Quantization as Histogram Segmentation: Optimal Scalar Quantizer Design in Network Systems

Dan Muresan and Michelle Effros, *Senior Member, IEEE*

***Abstract*—An algorithm for scalar quantizer design on discrete-alphabet sources is proposed. The proposed algorithm can be used to design fixed-rate and entropy-constrained conventional scalar quantizers, multiresolution scalar quantizers, multiple description scalar quantizers, and Wyner–Ziv scalar quantizers. The algorithm guarantees *globally optimal* solutions for conventional fixed-rate scalar quantizers and entropy-constrained scalar quantizers. For the other coding scenarios, the algorithm yields the best code among all codes that meet a given convexity constraint. In all cases, the algorithm run-time is polynomial in the size of the source alphabet. The algorithm derivation arises from a demonstration of the connection between scalar quantization, histogram segmentation, and the shortest path problem in a certain directed acyclic graph.**

***Index Terms*—Optimal design, multiple descriptions, multiresolution, scalar quantizer, successive refinement, Wyner–Ziv.**

I. Introduction

ANY lossy source code can be described as a quantizer followed by a lossless code. The quantizer maps the space of possible data values to a collection of allowed reproduction values, and the lossless code maps the space of allowed reproduction values to some uniquely decodable family of binary descriptions. This paper treats the problem of quantizer design.

A quantizer consists of a set of reproduction values and a mapping from the source alphabet to the reproductions. Each reproduction value is called a *codeword*, and the set of reproductions is called a *codebook*. The set of alphabet symbols that are mapped to a certain codeword is called the *codecell* associated with that codeword. The set of all codecells forms a partition of the source alphabet. From a communications system point of view, the codecells define a quantizer encoder, and the codewords define a quantizer decoder. Together, the quantizer encoder and decoder define the quantizer. We focus on scalar quantizers, where the codewords are scalar source reproduction values and the codecells partition the scalar alphabet.

In designing a quantizer, one can first design the decoder (the codewords) and then define the encoder (the codecells) with respect to the decoder, or one can design the encoder and then define the decoder with respect to the encoder. Most design algorithms take what we call a "codewords-first" approach—focusing on codebook design and defining the codecells as functions of the codebook. In this paper, we explore the alternative approach.

Our main contribution is a low-complexity scalar quantizer design algorithm for finite-alphabet sources. (The algorithm can also be used to design scalar quantizers for continuous-alphabet sources using either a finite training set or a discretization of the continuous alphabet.) The algorithm applies to a variety of types of scalar quantizers and in some cases guarantees a globally optimal design. The code types treated include fixed- and variable-rate (entropy-constrained) scalar quantizers for scenarios with a single encoder and one or more decoders. Examples include conventional scalar quantizers, multiresolution scalar quantizers, multiple-description scalar quantizers, Wyner–Ziv scalar quantizers, and any combination of these.

In each coding scenario, the algorithm finds the best code among all possible codes with convex codecells. Under mild constraints on the distortion measure, there always exists a globally optimal conventional scalar quantizer with convex codecells; thus, our algorithm yields globally optimal fixed- and variable-rate conventional scalar quantizers. In multiresolution, multiple-description, and Wyner–Ziv scalar quantization, there may or may not exist a globally optimal code with convex codecells; thus, our algorithm does not guarantee a globally optimal solution in these cases.

The runtime for our algorithm is polynomial in the size of the source alphabet. The order of the polynomial varies with the coding scenario. In addition to the optimal convex-codecell design, we also present a fast variation on the code design algorithm; the fast algorithm improves runtime (at the expense of some rate-distortion performance) by reducing the effective alphabet size. (That such a complexity savings is possible is not immediately obvious since direct alphabet reduction is suboptimal in general.)

Previous work in low-complexity, globally optimal scalar quantizer design treats fixed-rate, conventional (single-encoder, single-decoder) scalar quantizers for finite-alphabet sources. In fixed-rate coding, the number of codewords is fixed, all codewords are described at the same rate, and the design goal is to minimize the code's expected distortion with respect to a given probability mass function (pmf) or training set. In his 1964 doctoral dissertation [1], Bruce describes a polynomial-time dynamic programming algorithm for globally optimal fixed-rate conventional scalar quantizer design. In [2], Sharma shows how to reduce the complexity of Bruce's algorithm by using Fibonacci heaps. Wu and Zhang [3], [4] further refine the

Manuscript received October 19, 2004; revised September 15, 2007. The material in this paper was presented at The Data Compression Conference, Snowbird, UT, March 2002.

D. Muresan was an undergraduate at the California Institute of Technology, Pasadena, and an M.S. student at Stanford University, Stanford, CA, when this work was performed (e-mail: muresan@alumni.caltech.edu).

M. Effros is with the Department of Electrical Engineering, MC 136-93, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: effros@caltech.edu).

Communicated by V. A. Vaishampayan, Associate Editor At Large.

Digital Object Identifier 10.1109/TIT.2007.911170

dynamic programming algorithm by incorporating a number of other optimization techniques.

Our results, which first appeared in [5]–[7], generalize the earlier techniques to allow variable-rate scalar quantizer design and to treat quantizers with more than one decoder and quantizers with side information at the decoder. In particular, we:

- develop a polynomial-time algorithm for designing globally optimal entropy-constrained scalar quantizers, giving the optimal tradeoff between rate and distortion for scalar codes;
- generalize the dynamic-programming strategy to a family of single-source multiple-receiver and side-information source coding applications including fixed- and variable-rate conventional, multiresolution, and multiple-description scalar quantizers with or without decoder side information—giving a single strategy that for each scenario yields codes that are optimal with respect to the convex codecell constraint;
- clarify the connection between quantization, segmentation, and the shortest path problem in a certain directed acyclic graph; and
- derive conditions under which the optimal convex-codecell code fails and succeeds in guaranteeing the same solution as the corresponding unconstrained optimization.

In [8], Dumitrescu and Wu describe an algorithm designed to minimize the weighted sum of distortions of *all* subtrees of a fixed-depth tree-structured scalar quantizer; the distortion of each subtree with the same number of leaves is weighted equally in the optimization. Like our algorithm (which was first presented in [5]), that algorithm applies a dynamic programming approach to convex-codecell multiresolution code design; the unusual performance measure in [8] makes it difficult to compare the outcomes of the two strategies.[1] In [9], Dumitrescu and Wu describe an algorithm similar to the entropy-constrained multiresolution code design algorithm presented in [5], [7]. The same authors later describe a fixed-rate multiresolution design algorithm in [10]. In [11]–[13], they consider fixed-rate multiple-description scalar quantizer design for the special case where there are only two descriptions and those descriptions have the same rate. Like this paper, all of the papers of Dumitrescu and Wu are restricted to convex-codecell design.

The remainder of the paper is organized as follows. Section II sets up the problem and gives a precise description of the convex codecell constraint. Sections III–VI describe the code design algorithm for conventional, multiple-description, multiresolution, and side-information scalar quantizers. Each section describes both the optimal algorithm and its fast variation and includes a proof of the algorithm's optimality (subject to the convex-codecell constraint) when the algorithm is applied to the full source alphabet. Section VII derives the algorithm runtime for each scenario. Section VIII summarizes prior results and derives new theorems on codecell convexity. Section IX contains experimental results.

[1]By [8, Proposition 1], their performance measure can be made to mimic ours only for a very restricted collection of parameters.

II. Problem Setup

Each code design aims to optimize the tradeoff between expected rates and distortions. The distortion measure $d(x,\hat{x})$ is assumed to be nonnegative and the source description is required to be uniquely decodable. The expectation is taken relative either to a known pmf on a finite alphabet or to a finite training set on an arbitrary alphabet. (In some special cases, we can also approximate the optimal codes for continuous alphabet sources; we treat these cases under our discussion of fast approximations.) Given this effective restriction to finite alphabets and the assumption that the alphabet is ordered, in the remainder of this work we refer to scalar source alphabet $\{x_1,\ldots,x_N\}$ containing $N<\infty$ symbols by the symbol indices $\{1,\ldots,N\}$. The full alphabet is written $\{1,\ldots,N\}$, and the histogram describing the pmf is denoted by $\{p[1],\ldots,p[N]\}$.

Once either a quantizer encoder or a quantizer decoder is designed, optimization of the remaining portion is straightforward. We focus on optimal encoder design for conventional, multiresolution, multiple-description, and Wyner–Ziv scalar quantizers. In each of these scenarios, designing an encoder is equivalent to designing one or more partitions on the source alphabet $\{1,\ldots,N\}$. Each element S_i of partition $\mathcal{P}=\{S_1,\ldots,S_{|\mathcal{P}|}\}$ of $\{1,\ldots,N\}$ describes a collection of source symbols given the same binary description (a codecell). A conventional scalar quantizer encoder requires one partition. An M-description scalar quantizer encoder uses M partitions, one for each description. An M-resolution scalar quantizer encoder similarly requires M partitions.

Codecell Convexity

Scalar quantizer Q has *convex* codecells if for each encoder partition $\mathcal{P}=\{S_1,\ldots,S_{|\mathcal{P}|}\}$ of Q there exists an increasing sequence of natural numbers $T=\{t_k\}_{k=0}^{|\mathcal{P}|}$ with $t_0=0, t_{|\mathcal{P}|}=N$, and $S_i=(t_{i-1},t_i]$. The scalar quantizer design algorithm proposed in this work finds, in each coding scenario, the optimal scalar quantizer among all scalar quantizers with convex codecells.

Fast Approximations

We consider both true partition optimization relative to source alphabet $\{1,\ldots,N\}$ and approximate optimization where partitions are constrained to a coarse grid. Prior work on locally optimal code design for such quantized source distributions appears in [14].

We achieve the coarse grid by dividing the symbols of $\{1,\ldots,N\}$ into $\hat{N}$ convex, nonoverlapping cells $\{(g_{k-1},g_k]\}_{k=1}^{\hat{N}}$ with $0=g_0<g_1<\cdots<g_{\hat{N}}=N$. The fast approximation algorithm finds the best partition on this fixed, ordered set of indivisible cells. Using this coarse grid, the codecell comprising cells $\{(g_{a-1},g_a],\ldots,(g_{b-1},g_b]\}$ is

$$C_{(a-1,b]}=\left\{x_n : n\in\bigcup_{k=a}^{b}(g_{k-1},g_k]\right\}=\{x_{g_{a-1}+1},\ldots,x_{g_b}\}.$$

Example 1: Given source alphabet $\{1,\ldots,16\}$, let $\{g_k\}_{k=0}^{4}=\{0,4,8,12,16\}$. This breaks the source alphabet into four indivisible cells: $G_1=(0,4]=\{1,\ldots,4\}, G_2=$

$(4,8] = \{5,\ldots,8\}$, $G_3 = (8,12] = \{9,\ldots,12\}, G_4 = (12,16] = \{13,\ldots,16\}$. The search for an optimal segmentation on symbols $\{1,\ldots,16\}$ is approximated by a search for an optimal partition on cells $\{G_1,\ldots,G_4\}$. One possible partition on this coarse grid is $\{\{G_1,\ldots,G_3\},\{G_4\}\}$. The corresponding codecells are $C_{(0,3]} = \{x_1,\ldots,x_{12}\}$ and $C_{(3,4]} = \{x_{13},\ldots,x_{16}\}$, giving partition $\{(0,12],(12,16]\}$ on the original alphabet. □

While there are 2^{N-1} convex-codecell partitions of the original alphabet, only $2^{\hat{N}-1}$ of these are compatible with the $\hat{N}$-cell grid. Thus, guaranteeing optimality requires $\hat{N} = N$ (with each alphabet symbol being allotted a separate grid cell), but using $\hat{N} < N$ allows for faster code design. All results in this work are described in terms of the most general case ($\hat{N} \leq N$); for optimal convex-codecell design, $\hat{N} = N$ is required.

It is tempting to believe that a fast approximation can be achieved by replacing pmf $\{p[1],\ldots,p[N]\}$ on alphabet $\{x_1,\ldots,x_N\}$ by a new pmf $\{q[1],\ldots,q[\hat{N}]\}$ on some alphabet $\{y_1,\ldots,y_{\hat{N}}\}$ with the hopes that the best convex-codecell scalar quantizer for $\{q[1],\ldots,q[\hat{N}]\}$ is an optimal fast approximation for the best convex-codecell scalar quantizer for $\{p[1],\ldots,p[N]\}$. For example, we might set

$$y_k = \sum_{i=g_{k-1}+1}^{g_k} p[i]x_i \quad \text{and} \quad q[k] = \sum_{i=g_{k-1}+1}^{g_k} p[i].$$

While code design for $\{q[1],\ldots,q[\hat{N}]\}$ is faster than code design for $\{p[1],\ldots,p[N]\}$, the best code for $\{q[1],\ldots,q[\hat{N}]\}$ fails to achieve the best convex-codecell scalar quantizer consistent with our grid cells in general. The impediment is that for some distortion measures, we cannot assign a single source symbol to each of the $\hat{N}$ grid cells in a manner that preserves the accuracy of the distortion calculations. (Code design for the squared-error distortion measure is a notable exception, as shown in [14].) Thus, it is sometimes necessary to use the full source alphabet when evaluating distortions. In cases where the source alphabet can be replaced by an $\hat{N}$-symbol coarse alphabet or cases where the distortion can be calculated analytically, the coarse grid approach is useful for approximating optimal code design for continuous alphabet sources.

Segmentation and Quantization

The performance of a scalar quantizer depends on the source values and probabilities of the elements in each codecell. Under the convexity assumption, each codecell is a single segment of the source alphabet. We therefore view the problem of code design as a signal segmentation problem on pmf $\{p[1],\ldots,p[N]\}$. We want to design a segmentation of $\{p[1],\ldots,p[N]\}$ that achieves an optimal tradeoff between expected rates and expected distortions, and we want the design algorithm to be fast.

Optimal signal segmentation can be solved as a single-source shortest path problem in a weighted directed acyclic graph (WDAG). The WDAG is designed so that each possible segment is represented by a single edge in the WDAG, and each path from the graph's source node to its terminal node represents a distinct segmentation.

We demonstrate how to build a WDAG for each type of scalar quantizer. We call this WDAG the *partial RD graph* since it describes the rate and distortion contributions made by each potential codecell. For conventional scalar quantizer design, the graph vertices are $\mathcal{V} = \{0,\ldots,\hat{N}\}$, the graph edges are $\mathcal{E} = \{(u,v) \in \mathcal{V}^2 : v > u\}$, the weight of edge (u,v) equals the contribution to the rate–distortion performance made by codecell $(u,v]$, and each path $\{(v_0,v_1),(v_1,v_2),\ldots,(v_{k-1},v_k)\}$ from node $v_0 = 0$ to node $v_k = N$ represents a distinct partition $\{C_{(v_0,v_1]}, C_{(v_1,v_2]},\ldots,C_{(v_{k-1},v_k]}\}$ of the source alphabet. For multiple-description and multiresolution scalar quantizers, the graph vertices and edges become a bit more complicated (see the corresponding sections for details), but there is again a one-to-one correspondence between the paths from the graph's single source to its single sink and the partitions that define the given form of scalar quantizer. In each scenario, we prove that optimal convex-codecell scalar quantizer design is equivalent to solving a single-source shortest path problem on that graph.

A number of authors have previously exploited the relationship between segmentation and shortest path problems to obtain rate–distortion optimality in source codes that rely on segmentation. In [15], Chou and Lookabough use dynamic programming to segment a source sequence for a variable-rate code with codewords of varying dimensions ("variable-to-variable" length coding). Effros, Chou, and Gray take a similar approach to signal segmentation for variable-dimension weighted universal coding in [16]. In [17], Xiong, Herley, Ramchandran, and Orchard use a dynamic programming algorithm for signal segmentation in a source code with time-varying wavelet packets; this algorithm is generalized and refined in [18]. Schuster and Katsaggelos use the directed acyclic graph (DAG) shortest path algorithm to separate objects from background for video coding in [19]. In each of these examples, segmentation is part of the encoding procedure, and the optimal segmentation is the segmentation that yields the best tradeoff between rate and distortion in the resulting code. These results differ fundamentally from our approach since they segment the sequence of source observations during encoding while we segment the source pmf during code design.

Single-Source Shortest Path Algorithms

Once we have built the partial RD graph, we use standard techniques to solve the shortest path problem on that WDAG. As outlined in [20], the single-source shortest path problem in a WDAG with V vertices and E edges can be solved in $O(V+E)$ time. The algorithm relies on the observation that if a shortest path from v_0 to v_l stops at intermediate vertex v_j, then its path from v_0 to v_j must also be a shortest path. Using this observation, the algorithm iteratively calculates a shortest path from v_0 to v_j for every j. Iteration i "relaxes" (improves) the shortest path estimate to each vertex v_j with $j \geq i$ by comparing the best path found so far to the path that traverses a shortest path from v_0 to v_i and then the edge (v_i,v_j).

The dynamic programming algorithm in [18] (not presented explicitly as a graph algorithm in that paper) has the same complexity but is simpler and more natural in that no relaxation is involved. Rather, in the nth step, a shortest path to the nth vertex

is computed (definitively) based on previous shortest paths. Precisely, a shortest path from v_0 to v_n must be a shortest path from v_0 to v_i $(i < n)$ followed by the edge (v_i, v_n). Comparing these alternatives (and relying on the previous calculation of a shortest path from v_0 to v_i for all $i < n$) allows the precise solution to the shortest path problem from v_0 to v_n.

The first step in both algorithms is to order the V vertices of the graph in a sequence $\{v_k\}_{k=1}^{V}$ such that if $k > \ell$, then there are no arcs from v_k to v_ℓ (this is always possible in an acyclic graph since the edge set induces a partial order on the vertex set).

The following algorithm takes a graph as its input and produces a topological order on the vertices of that graph in$O(V + E)$ steps. We number the vertices according to the order in which they are placed in some sorted list S, which is initialized to be empty. For each vertex v, keep a counter $a[v]$, initialized with the *in-degree* of v (the number of incoming arcs). We will decrement $a[v]$ each time one of its predecessors is added to S. When $a[v]$ becomes 0, vertex v becomes eligible for addition to list S since all of its predecessors have already been numbered. We therefore maintain a stack of "free vertices" F initialized with the set of vertices with in-degree zero. At each time $i \geq 1$, remove a vertex v from stack F, append it to the sorted sequence S, and decrement $a[u]$ for all successors u of v. Whenever some $a[u]$ becomes zero, place u on F. The algorithm stops when S contains all V vertices; if F becomes empty at any point before S has filled up, the graph has a cycle. This problem cannot arise in our algorithm since our graphs are acyclic by construction; thus, our algorithm is guaranteed to run to completion.

Given the sorted vertex sequence $\{v_k\}_{k=1}^{V}$, the dynamic programming algorithm proceeds as follows. At each step $\ell \in \{1, \ldots, V\}$, compute the weight of a shortest path to vertex v_ℓ as

$$s[\ell] = \min(s[k] + w_{k,\ell})$$

where $w_{k,\ell}$ is the weight on arc (v_k, v_ℓ) and the minimization is performed over the set of indices k for which (v_k, v_ℓ) is in our edge set; $s[\ell] = 0$ if there are no such indices k. Since the sequence $\{v_k\}_{k=1}^{V}$ is sorted topologically, if (v_k, v_ℓ) is an edge, then $k < \ell$ and $s[k]$ is known when we calculate $s[\ell]$.

III. Conventional Scalar Quantization

We begin by considering the conventional source coding paradigm, where a single encoder describes information to be interpreted by a single decoder. Since the solution to the fixed-rate problem appears in [1]–[4], this section treats only variable-rate code design.

Optimization Criterion

As discussed in Section II, we describe the encoder of a scalar quantizer by the partition $\mathcal{P}$ that defines its codecells. Let $D(\mathcal{P})$ and $R(\mathcal{P})$ be the expected distortion and output entropy, respectively, for the variable-rate scalar quantizer corresponding to partition $\mathcal{P}$. (Measuring rate as entropy in variable-rate codes separates the code design from the specific entropy code implementation.) For any partition achieving a point on the operational distortion-rate function $\hat{D}(R) = \inf_{\mathcal{P}}\{D(\mathcal{P}) : R(\mathcal{P}) \leq R\}$, the corresponding variable-rate scalar quantizer is optimal in the sense that no other partition can achieve a lower distortion using the same or lower entropy. For any $\mathcal{P}$ with $(R(\mathcal{P}), D(\mathcal{P}))$ lying on the lower convex hull of $\hat{D}(R)$, there exists a Lagrangian multiplier $\lambda > 0$ for which $\mathcal{P}$ minimizes

$$J(\mathcal{P}, \lambda) = D(\mathcal{P}) + \lambda R(\mathcal{P}). \tag{1}$$

We therefore use the Lagrangian $J(\mathcal{P}, \lambda)$ as the optimization criterion for our partition design. The resulting code is called an entropy-constrained scalar quantizer (ECSQ) [21].

The key observation is that rate and distortion (and thus the Lagrangian cost) are additive over codecells. That is, for a partition $\mathcal{P}$, $R(\mathcal{P}) = \sum_{C\in\mathcal{P}} r(C)$ and $D(\mathcal{P}) = \sum_{C\in\mathcal{P}} d(C)$. Here $r(C) = -p(C)\log p(C)$ is called the *partial rate* of codecell C,[2] and $d(C) = \sum_{n\in C} p[n]d(x_n, \mu(C))$ is called the *partial distortion* of codecell C; $p(C) = \sum_{n\in C} p[n]$ and $\mu(C)$ are the probability and optimal codeword for codecell C. (The optimal codeword is the reproduction value that minimizes $d(C)$; this value is not necessarily in the finite source alphabet. For example, when $d(x, \hat{x}) = (x - \hat{x})^2$, the optimal codeword $\mu(C)$ is the centroid $\mu(C) = \sum_{n\in C} p[n]x_n/p(C)$ of codecell C.) The additivity of our Lagrangian performance measure $J(\mathcal{P}, \lambda)$ over codecells is critical for defining the partial RD graph.

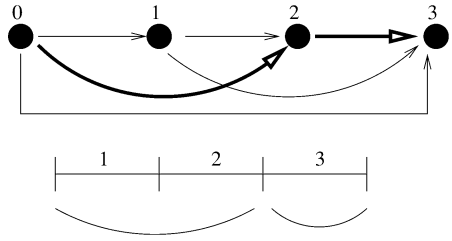

Fig. 1. Partial RD graph for an alphabet with $\hat{N} = 3$ cells. The path describing the partition $\{(0, 2], (2, 3]\}$ appears in bold.

Partial RD Graph

The partial RD graph for ECSQ is shown in Fig. 1. The vertex set is $\mathcal{V} = \{0, \ldots, \hat{N}\}$. For every vertex pair (u, v) with $u < v$ there is an *arc* (directed edge) from u to v. The arc corresponds to a codecell $C_{(u,v]}$ comprising coarse alphabet cells $\{G_{u+1}, \ldots, G_v\}$. The weight $w_{u,v}$ of arc (u, v) is the Lagrangian cost $d(C_{(u,v]}) + \lambda r(C_{(u,v]})$ for codecell $C_{(u,v]}$.

Each path $\{(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)\}$ from node $v_0 = 0$ to node $v_k = \hat{N}$ represents a distinct partition

$$\mathcal{P} = \{C_{(v_0,v_1]}, C_{(v_1,v_2]}, \ldots, C_{(v_{k-1},v_k]}\}$$

of the source alphabet. The total weight of path $\{(v_{i-1}, v_i)\}_{i=1}^{k}$ is

$$\begin{aligned}\sum_{i=1}^{k} \left[d(C_{(v_{i-1},v_i]}) + \lambda r(C_{(v_{i-1},v_i]})\right] &= \sum_{i=1}^{k} d(C_{(v_{i-1},v_i]}) + \lambda \sum_{i=1}^{k} r(C_{(v_{i-1},v_i]}) \\ &= D(\mathcal{P}) + \lambda R(\mathcal{P}).\end{aligned}$$

Thus, the shortest path from node 0 to node $\hat{N}$ describes the partition with the optimal Lagrangian performance. Before proceeding to a formal proof of this assertion, we illustrate these ideas with an example.

[2]All logarithms in this paper are base-2.

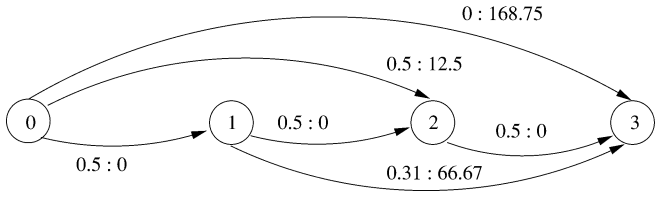

Fig. 2. The 1-DSQ (description scalar quantization) partial RD graph for the source in Example 2. Edge (u, v) is labeled as $r(C_{(u,v]}) : d(C_{(u,v]})$ to show the rate and distortion of the corresponding codecell.

Example 2: As a simple example, we design a conventional ECSQ for a source with a 3-symbol alphabet ($\hat{N} = N = 3$) and the squared-error distortion measure. Our source alphabet is $x_1 = 0, x_2 = 10, x_3 = 30$ with pmf $p[1] = p[2] = 1/4$, and $p[3] = 1/2$.

We begin by computing the partial rates and distortions on the partial RD graph for this source. Arc (u, v) corresponds to codecell $C = (u, v]$ which has partial rate $r(C) = -p(C) \log p(C)$ and partial distortion $d(C) = \sum_{n \in C} p[n] d(x_n, \mu(C))$. For example, arc $(0, 2)$ represents the codecell $C_{(0,2]}$ which contains source symbols $x_1 = 0$ and $x_2 = 10$. The probability of this codecell is $p(C) = p[1] + p[2] = 1/2$, and the cell centroid is $\mu(C) = (0 + 10)/2 = 5$, giving $r(C) = -(1/2) \log(1/2) = 1/2$ and $d(C) = (1/4)(0-5)^2 + (1/4)(10-5)^2 = 12.5$. Fig. 2 shows the partial rate and distortion for each arc.

When $\lambda = 100, w_{u,v} = d(C_{(u,v]}) + 100 r(C_{(u,v]})$. Thus, $w_{0,1} = w_{1,2} = w_{2,3} = 50$, $w_{0,2} = 62.5, w_{1,3} = 97.80$, $w_{0,3} = 168.75$. The shortest path to node 1 has weight $w_{0,1} = 50$; the shortest path to node 2 has weight $\min\{50 + w_{1,2}, w_{0,2}\} = 62.5$; the shortest path to node 3 has weight $\min\{50 + w_{1,3}, 62.5 + w_{2,3}, w_{0,3}\} = 112.5$. This shortest path is achieved by partition $\mathcal{P} = \{C_{(0,2]}, C_{(2,3]}\}$; thus, the optimal ECSQ for $\lambda = 100$ breaks the source alphabet into codecells $\{0, 10\}, \{30\}$. The code achieves entropy 1.0 bit per symbol and expected squared-error distortion 12.5.

Proof of Correctness

We now formally prove that optimal quantizer design is equivalent to finding the shortest path between vertices 0 and $\hat{N}$ in the partial RD graph. We do this by first demonstrating a one-to-one correspondence between partitions and paths in the graph. We then show that the total weight on a path from 0 to $\hat{N}$ equals the Lagrangian for the partition corresponding to that path.

A path of length l from 0 to $\hat{N}$ is a chain of edges $(v_0, v_1), (v_1, v_2), \ldots, (v_{l-1}, v_l)$ with $v_0 = 0$ and $v_l = \hat{N}$. Because of the structure of our WDAG, (v_i, v_{i+1}) is an arc if and only if $v_i < v_{i+1}$. Therefore, we can associate bijectively to each arc (v_i, v_j) the codecell $(v_i, v_j]$. For a partition, consecutive codecells must be adjacent; for a path, consecutive arcs must share one vertex. Therefore, an enumeration of the arcs corresponding to the codecells in a partition forms a path, and *vice versa*. This establishes the required two-way correspondence.

The equivalence between the total weight on a path from 0 to $\hat{N}$ and the Lagrangian for the partition corresponding to the path follows directly from our construction. The weight on edge (v_i, v_j) is the Lagrangian cost $d(C_{(v_i,v_j]}) + \lambda r(C_{(v_i,v_j]})$. The weight of a path $(v_0, v_1)(v_1, v_2), \ldots, (v_{l-1}, v_l)$ is the sum of the weights of its edges

$$\begin{aligned}\sum_{i=1}^{l} & d(C_{(v_{i-1},v_i]}) + \lambda r(C_{(v_{i-1},v_i]}) \\ &= \left[\sum_{i=1}^{l} d\left(C_{(v_{i-1},v_i]}\right) + \lambda \sum_{i=1}^{l} r\left(C_{(v_{i-1},v_i]}\right)\right] \\ &= D(\mathcal{P}) + \lambda R(\mathcal{P})\end{aligned}$$

where $\mathcal{P} = \{C_{(v_0,v_1]}, C_{(v_1,v_2]}, \ldots, C_{(v_{l-1},v_l]}\}$ is a unique partition of $\{1, \ldots, \hat{N}\}$.

Complexity

The partial RD graph has $\hat{N}+1$ vertices and $\hat{N}(\hat{N}+1)/2$ arcs. The vertex labels give the topological order of the vertices, so in this case no vertex ordering is required in the shortest path algorithm. Building the partial RD graph requires finding each of the $O(\hat{N}^2)$ weights. Each weight calculation requires $O(N)$ operations in general. Thus, the graph construction takes $O(\hat{N}^2 N)$ time. When $d(x, \hat{x}) = (x - \hat{x})^2$, this can be reduced to $O(N)$ time by computing the cumulative moments of order 0, 1, and 2 for the source pmf so that the partial rates and distortions can be computed using differences of these cumulative moments [3]. Likewise, when $d(x, \hat{x})$ is monotonic in $|x - \hat{x}|$, the graph construction can be reduced to $O(N^2)$ [4]. (This result cannot take advantage of a coarse grid, so we set $\hat{N} = N$ in this case.) Given the partial RD graph, the single-source shortest path problem can be solved in $O(\hat{N}^2)$ time. Performing the weight calculations during the shortest path algorithm (rather than in an independent preprocessing step) yields a total complexity for globally optimal ECSQ design of $O(\hat{N}^2 + N), O(N^2)$, or $O(\hat{N}^2 N)$, depending on the distortion measure.

IV. Multiple-Decoder Systems

The algorithm described in Section III treats conventional scalar quantizer design—that is, scalar quantization for the source coding paradigm, where a single encoder describes information to be decoded by a single decoder. We next generalize that algorithm to scenarios where a single source is described to multiple decoders. Examples of single-source, multiple-decoder systems include multiresolution and multiple-description scalar quantizers.

Multiresolution scalar quantization (MRSQ) [22]–[26] produces a binary source description that can be described in increments. Reading only the first increment gives a reproduction of the poorest quality; each subsequent increment improves the quality of the source reproduction. The increments have a natural order, and each incremental description can only be decoded if all preceding increments have been received before it. Since the binary description may be decoded at a variety of rates, the MRSQ decoder effectively contains a collection of decoders, one for each rate at which the binary sequence may be decoded. MRSQ is useful in applications where the compressed data sequence is intended for multiple users with differing rate and reproduction quality needs.

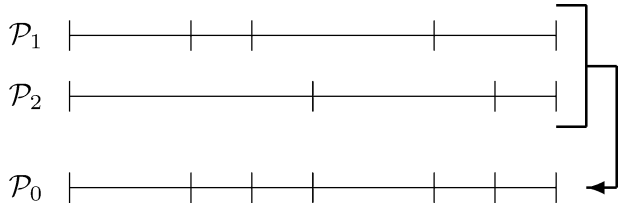

Fig. 3. Side partitions $\mathcal{P}_1$ and $\mathcal{P}_2$ and central partition $\mathcal{P}_0$.

Multiple-description scalar quantization (MDSQ) [27]–[30] is another single-source, multiple-decoder scalar quantization system. In this case, the source description is broken into a collection of packets, and reconstruction under any combination of received and lost packets is required. MDSQ is useful in diversity systems, where the encoder sends separate descriptions of the source over M different channels and reconstruction is required using the successfully received descriptions. MDSQ is also useful for packet-based communication systems and distributed data storage systems, where reconstruction of the data using a subset of the original set of source descriptions may be required.

In both MRSQ and MDSQ, the increments or packets are assumed to be labeled, so that the decoder can distinguish which increments or packets it receives.

We next describe the optimal code design algorithm for single-source, multiple-decoder systems. We focus our description on variable-rate MDSQ design with $M = 2$ (thus 2DSQ instead of MDSQ) for simplicity. The solutions for fixed-rate coding and $M > 2$ are natural extensions of the variable-rate $M = 2$ solution, as described at the end of this section. The optimal MDSQ design algorithm leads to an optimal MRSQ design algorithm since MRSQ is a special case of MDSQ where decoders are designed for a *subset* of the possible packet-loss scenarios rather than all packet-loss scenarios. We can, however, achieve lower rates for some types of MDSQ with restricted packet-loss scenarios. MRSQ is a prime example, as discussed in Section V.

Optimization Criterion

The encoder of a 2DSQ gives two separate data descriptions and is therefore defined by two distinct partitions, $\mathcal{P}_1$ and $\mathcal{P}_2$, of the source alphabet. The codecells of each partition are convex by assumption. The decoder may receive either of the two descriptions alone or both descriptions together. When the decoder receives only a single description, say the index of codecell C_i in partition $\mathcal{P}_i$, the decoder knows that the source symbol x satisfies $x \in C_i$. When the decoder receives two descriptions, it knows that the source symbol x satisfies $x \in C_1 \cap C_2$ for some codecells $C_1 \in \mathcal{P}_1$ and $C_2 \in \mathcal{P}_2$, yielding an effective underlying partition $\mathcal{P}_0 = \{C_1 \cap C_2 : C_1 \in \mathcal{P}_1, C_2 \in \mathcal{P}_2\}$ of the data. Partition $\mathcal{P}_0$ is called the *central partition*, while $\mathcal{P}_1$ and $\mathcal{P}_2$ are called *side partitions*. Fig. 3 shows an example. Like $\mathcal{P}_1$ and $\mathcal{P}_2$, $\mathcal{P}_0$ has convex codecells.

Since either description of a 2DSQ may be received without the other, each partition must be uniquely decodable on its own, and the rate required to describe partition i is $R_i = R(\mathcal{P}_i)$ $(i \in \{1,2\})$. The distortion for partition i is $D_i = D(\mathcal{P}_i)$ $(i \in \{0,1,2\})$. Thus, (R_i, D_i) $(i \in \{1,2\})$ is the expected rate–distortion performance when only the ith description is received, and D_0 is the expected distortion when both descriptions are received. (Rate $R(\mathcal{P}_0)$ is not used since partition $\mathcal{P}_0$ is described only through the description of $\mathcal{P}_1$ and $\mathcal{P}_2$.) Again $D(\mathcal{P}_i) = \sum_{C \in \mathcal{P}_i} d(C)$ and $R(\mathcal{P}_i) = \sum_{C \in \mathcal{P}_i} r(C)$, with $d(C) = \sum_{n \in C} p[n] d(x_n, \mu(C))$ and $r(C) = -p(C) \log p(C)$ as defined in Section III.

Let

$$\mathcal{D}^{\text{vr}} = \left\{ \left(R(\mathcal{P}_i)|_{i=1}^2, D(\mathcal{P}_i)|_{i=0}^2 \right) : \mathcal{P}_1 \text{ and } \mathcal{P}_2 \text{ induce } \mathcal{P}_0 \right\}$$

be the set of rate–distortion vectors achievable through variable-rate coding. Given a pair of partitions $(\mathcal{P}_1, \mathcal{P}_2)$ achieving a variable-rate coding point $(R_1, R_2, D_0, D_1, D_2)$, $(\mathcal{P}_1, \mathcal{P}_2)$ is optimal if there does not exist another pair of partitions $(\mathcal{P}_1', \mathcal{P}_2')$ with $R(\mathcal{P}_i') \le R_i$ for all $i \in \{1,2\}$ and $D(\mathcal{P}_i') \le D_i$ for all $i \in \{0,1,2\}$ for which at least one of the inequalities is strict. For any $(\mathcal{P}_1, \mathcal{P}_2)$ with $(R(\mathcal{P}_i)|_{i=1}^2, D(\mathcal{P}_i)|_{i=0}^2)$ on the lower convex hull of $\mathcal{D}^{\text{vr}}$, there exist nonnegative Lagrangian vectors $\nu^3 = (\nu_0, \nu_1, \nu_2)$ and $\lambda^2 = (\lambda_1, \lambda_2)$ for which $(\mathcal{P}_1, \mathcal{P}_2)$ minimizes

$$J^{\text{vr}}((\mathcal{P}_1, \mathcal{P}_2), \nu^3, \lambda^2) = \nu_0 D_0 + \sum_{i=1}^{2} [\nu_i D(\mathcal{P}_i) + \lambda_i R(\mathcal{P}_i)].$$

We therefore use $J^{\text{vr}}((\mathcal{P}_1, \mathcal{P}_2), \nu^3, \lambda^2)$ as the optimization criterion for partition design in variable-rate 2DSQs.

Unfortunately, $J^{\text{vr}}((\mathcal{P}_1, \mathcal{P}_2), \nu^3, \lambda^2)$ is a function of $D(\mathcal{P}_0)$ and $\mathcal{P}_0$ varies with both partitions. As a result, we cannot divide this optimization into separate optimizations of $\mathcal{P}_1$ and $\mathcal{P}_2$. Instead, a joint optimization is required, which complicates our design of the partial RD graph.

Partial RD Graph

Our goal is to design a partial RD graph with a one-to-one correspondence between paths through the RD graph and partition pairs. The total weight of a path should be the Lagrangian cost $J^{\text{vr}}((\mathcal{P}_1, \mathcal{P}_2), \nu^3, \lambda^2)$ of the corresponding partition pair. If we can build such a graph, then running a shortest path algorithm on this graph guarantees an optimal 2DSQ.

To begin, we define the vertex set as $\mathcal{V} = \{0, \ldots, \hat{N}\}^2$. This collection will be reduced in the discussion that follows. In the partial RD graph for conventional scalar quantization, the paths from vertex 0 to vertex v specify all partitions that span the cell range $\{1, \ldots, v\}$. Similarly, in the MDSQ partial RD graph, the paths from vertex $(0,0)$ to vertex (v_1, v_2) represent all *partition pairs* $(\mathcal{P}_1, \mathcal{P}_2)$ such that $\mathcal{P}_1$ spans $\{1, \ldots, v_1\}$ and $\mathcal{P}_2$ spans $\{1, \ldots, v_2\}$ (we ignore the central partition for the time being). For example, the paths from vertex $(0,0)$ to vertex $(\hat{N}, \hat{N})$ designate all partition pairs for the full source alphabet. Note that the partitions in a pair are always aligned on the left side.

Construction of the edge set presents a certain difficulty; naively, there should be an arc from vertex (v_1, v_2) to all vertices (v_1', v_2) and (v_1, v_2') such that $v_1' > v_1$ and $v_2' > v_2$, as shown in the example in Fig. 4, since we can extend a partition pair by adding a codecell to either of the two side partitions. (In

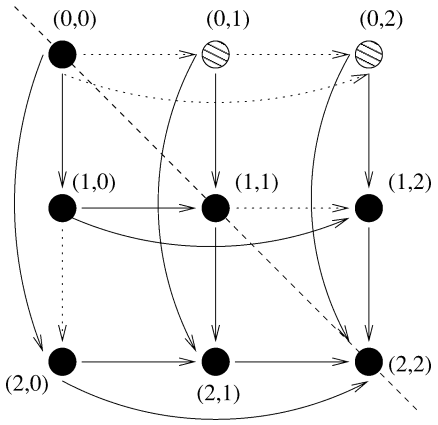

Fig. 4. Partial RD graph, two descriptions, 2×2 cells.

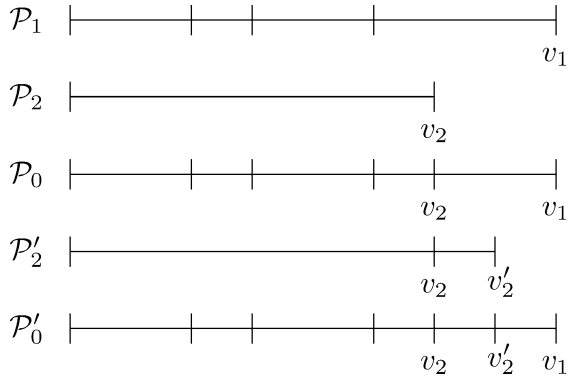

Fig. 5. The central partition is determined only to the end of the shorter side partition. In this example, the central partition changes in $(v_2, v_1]$ when cell $(v_2, v_2']$ is added to $\mathcal{P}_2$. The central partition on values less than $\min\{v_1, v_2\}$ cannot change with cell additions.

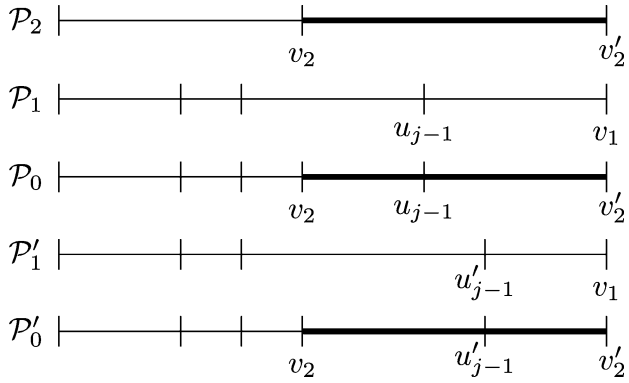

Fig. 6. The shorter side partition is too short. In this example, $v_2 < u_{j-1}$ implies that the central partition on $(v_2, v_1]$ (and thus the weight on the arc from (v_1, v_2) to (v_1, v_2')) varies with u_{j-1}.

the discussion that follows, we remove the shaded vertices and dotted arcs from Fig. 4, but for now they should be treated like their solid counterparts.) We emphasize the fact that only one side partition is extended at a time, thus each arc corresponds to a single codecell being added to one of the side partitions. Extending one side partition by one codecell may cause one or possibly multiple codecells to be added to the central partition. To be able to assign a unique weight to the corresponding arc, the total Lagrangian cost of the newly added codecells must be a function only of the labels of the two vertices that the given arc joins. The codecell added to one of the side partitions is clearly determined by the two vertex labels only, but the central codecells may not be. The problem arises from uncertainty about the central partition as shown by the following example.

When the side partitions have equal length (i.e., for any vertex (v, v)) computing $\mathcal{P}_0$ from $\mathcal{P}_1$ and $\mathcal{P}_2$ is straightforward (see Fig. 3). When the side partitions have different lengths, however, it is not clear how to determine the codecells in the central partition. For example, consider the (v_1, v_2) partition pair with $v_1 > v_2$ shown in Fig. 5. Naively, we would find the codecells of $\mathcal{P}_0$ by merging the list of codecell thresholds; by this procedure, the last central codecell would be $(v_2, v_1]$. Notice, however, that adding a new side codecell $(v_2, v_2']$ to $\mathcal{P}_2$, as illustrated in the lower portion of Fig. 5, breaks the central codecell $(v_2, v_1]$ into two new central codecells $(v_2, v_2']$ and $(v_2', v_1]$ that depend on v_2'. Thus, central codecells containing $\{1, \ldots, \min(v_1, v_2)\}$ are not influenced by the subsequent addition of side codecells, but central codecells containing $\{\min(v_1, v_2) + 1, \ldots, \max(v_1, v_2)\}$ may vary as a function of future additions to the shorter side partition.

The uncertainty about $\mathcal{P}_0$ demonstrated in the previous example threatens our ability to devise a unique arc labeling. For example, consider the situation shown in Fig. 6. In this example, $\mathcal{P}_1$ is a partition spanning $\{1, \ldots, v_1\}$ and $\mathcal{P}_2$ is initially a partition spanning $\{1, \ldots, v_2\}$. We extend the shorter partition $\mathcal{P}_2$ by adding codecell $(v_2, v_2']$. The figure shows two examples of possible values for partition $\mathcal{P}_1$. The value of v_1 is the same in both examples, so the Lagrangian cost of extending $\mathcal{P}_2$ must in both cases be captured in the weight of the edge from (v_1, v_2) to (v_1, v_2'). Unfortunately, the central codecells created by the extension are different in the two cases, and thus no single weight can actually capture the Lagrangian cost of the given arc. This problem seems to arise from the fact that $\mathcal{P}_2$ is "too short"—one and a half codecells shorter than $\mathcal{P}_1$. We need a way to keep side partitions more "even" to avoid this type of problem.

The *codecell lag*, defined using the threshold sequence $\{t_k\}_{k=0}^{J}$ for partition $\mathcal{P}_1 = \{C_{(t_0,t_1]}, \ldots, C_{(t_{J-1},t_J]}\}$ and $\{u_k\}_{k=0}^{K}$ for partition $\mathcal{P}_2 = \{C_{(u_0,u_1]}, \ldots, C_{(u_{K-1},u_K]}\}$ provides a measure of how "even" two partitions are; partitions must be aligned on the left side ($t_0 = u_0$) for the following definition to apply. We say that $\mathcal{P}_2$ lags $\mathcal{P}_1$ by L codecells if there are L codecells in $\mathcal{P}_1$ not lying entirely inside the range covered by $\mathcal{P}_2$ (i.e., $t_{K-L} \leq u_J < t_{K-L+1}$). We want to keep the codecell lag between the two side partitions at most 1. (A lag of zero means that the partitions are of equal total length $t_J = u_K$, a case which we handle later in this section.) Call a partition pair with codecell lag less than or equal to 1 *valid*.

Suppose that $\mathcal{P}_2$ is the shorter partition $(u_K < t_J)$. Then by definition, a lag of 1 implies that the range $\{u_K + 1, \ldots, t_J\}$ is included in the last codecell of $\mathcal{P}_1$ (i.e., $t_{J-1} < u_K < t_J$). Adding codecell $(u_K, u_{K+1}]$ to $\mathcal{P}_2$ results in a single central codecell $(u_K, \min(t_J, u_{K+1})]$ being generated, regardless of the structure of the side partitions; therefore, we achieve our objective. The weight on each arc is defined as the Lagrangian cost associated with one new side codecell and one new central codecell.

We now have a condition (validity) that guarantees that adding side codecells results in predictable central codecells being generated; we next constrain the partial RD graph to obey that condition. We want to keep partition pairs with codecell lag more than 1 unrepresented in the vertices of the graph. Since the partition pairs represented in each vertex are defined by the paths between $(0,0)$ and that vertex, it is enough to break certain malignant paths by removing arcs from the graph.

Our goal is accomplished by eliminating arcs that extend the longer side partition in a partition pair: extending the longer side partition always increases codecell lag by 1, thus creating undesirable partitions with a lag of 2 or more; by contrast, extending the shorter side partition only increases lag when $v_1 = v_2$ (the exact change in lag depends on the exact structure of the partitions), and changes lag from 0 to 1 when $v_1 = v_2$, therefore (by an inductive argument), lag can be kept at 1 or 0. When $v_1 = v_2$, no central codecells are added regardless of which side partition is extended and by how much. To avoid building more than one path for a single pair of partitions, we choose to extend $\mathcal{P}_1$ if $v_1 = v_2$.

The result is a *partition extension rule* that allows only two types of arcs: $((v_1, v_2), (v_1, v_2'))$ with $v_1 > v_2, v_2' > v_2$ and $(((v_1, v_2), (v_1', v_2))$ with $v_1 \leq v_2, v_1' > v_1$. In the example of a partial RD graph for 2DSQ given in Fig. 4, the disallowed arcs are dotted. A consequence of eliminating these arcs is that some vertices become inaccessible from $(0,0)$ and therefore cannot be visited; we eliminate these vertices and the arcs originating from them. (To be precise, the inaccessible vertices are exactly those of the form $(0, v_2)$ with $v_2 > 0$; they are shaded in Fig. 4.)

After removing inaccessible vertices, we are left with a graph of $(\hat{N}+1)^2 - \hat{N}$ vertices. To count arcs, we divide vertices (v_1, v_2) in two categories: those with $v_1 = v_2$ and those with $v_1 \neq v_2$. If $v_1 = v_2 = v$ ($v \in \{0, \ldots, \hat{N}\}$), vertex (v_1, v_2) has $\hat{N} - v$ outgoing arcs; thus the graph contains a total of $\hat{N}(\hat{N}+1)/2$ arcs from the first category. In the second category, either $v_1 < v_2$ or $v_2 < v_1$. If $v_1 < v_2$, vertex (v_1, v_2) has $\hat{N} - v_1$ outgoing arcs; for each $v_1 \in \{0, \ldots, \hat{N}\}$ there are $\hat{N} - v_1$ such vertices with $v_2 \in \{v_1 + 1, \ldots, \hat{N}\}$; thus, if we had not removed inaccessible vertices, the graph would contain

$$\sum_{v=0}^{\hat{N}} (\hat{N} - v)^2 = \hat{N}(\hat{N}+1)(2\hat{N}+1)/6$$

second-category arcs from vertices with $v_1 < v_2$ and an equal number from vertices with $v_2 < v_1$. We therefore bound the total number of arcs by $\hat{N}(\hat{N}+1)(4\hat{N}+5)/6$ (removing the inaccessible vertices eliminates $\hat{N}^2$ of these edges).

The weight W on an arc $((v_1, v_2), (v_1', v_2'))$ is as follows.

- When $v_1 = v_2 = v_2' = v$, one codecell $C_1 = C_{(v, v_1']}$ is added to $\mathcal{P}_1$ and none are added to $\mathcal{P}_0$; thus, $W = \nu_1 d(C_1) + \lambda_1 r(C_1)$.
- When $v_2 = v_2' > v_1$, one codecell $C_1 = C_{(v_1, v_1']}$ is added to $\mathcal{P}_1$ and one central codecell $C_0 = C_{(v_1, \min(v_1', v_2))]}$ is generated; thus, $W = \nu_0 d(C_0) + \nu_1 d(C_1) + \lambda_1 r(C_1)$.
- When $v_1 = v_1' > v_2$, one codecell $C_2 = C_{(v_2, v_2']}$ is added to $\mathcal{P}_2$ and one central codecell $C_0 = C_{(v_2, \min(v_1, v_2'))]}$ is generated; thus, $W = \nu_0 d(C_0) + \nu_2 d(C_2) + \lambda_2 r(C_2)$.

When $d(x, \hat{x}) = (x - \hat{x})^2$ and we pre-compute the cumulative moments as in [3], each weight can be computed in constant time since our construction guarantees that at most two codecells are generated by adding an arc to a given path.

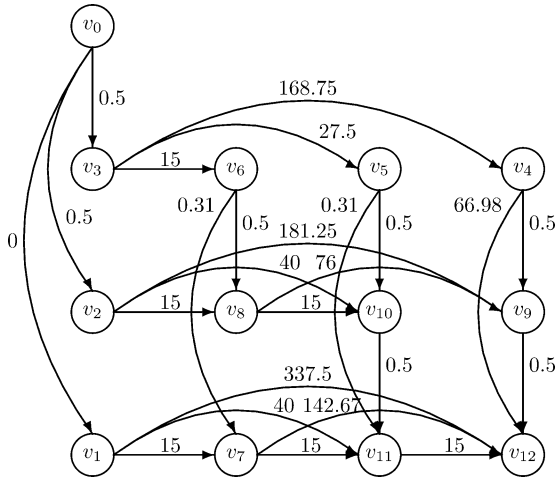

Fig. 7. The partial RD graph for the 2DSQ in Example 3; inaccessible vertices have been removed.

This completes the description of the 2DSQ partial RD graph. The graph design ensures a one-to-one correspondence between paths and partitions, and each path weight equals the Lagrangian cost of the corresponding partition. Thus, running one of the shortest path algorithms described in Section III with source $(0,0)$ and sink $(\hat{N}, \hat{N})$ yields an optimal 2DSQ among all 2DSQs with convex codecells. We again work a simple example before tackling a formal proof of this correspondence.

Example 3: We design a 2DSQ for the source considered in Example 2. We set $\lambda_1 = 1, \lambda_2 = 30, \nu_1 = 0$, and $\nu_2 = 1$ for the side partitions, and $\nu_0 = 3$ for the central partition.

We begin by sorting the vertices of the 2DSQ graph shown in Fig. 7. The only requirement for this list is that if there exists an edge from u to v, then vertex u comes before v in the sorted list. Many possible orderings meet this constraint. The sorted list created by our algorithm is $(0,0), (3,0), (2,0), (1,0), (1,3), (1,2), (1,1), (3,1), (2,1), (2,3), (2,2), (3,2), (3,3)$. For each $k \in \{0, \ldots, 12\}$, let v_k be the kth vertex in this list.

Next, we calculate the weights for graph edges. For each edge, we compute the corresponding side and intersection codecells and their weighted partial rates and distortion. For example, the arc from $v_3 = (1,0)$ to $v_5 = (1,2)$ generates codecell $(0,2]$ in side-partition 2 and codecell $(0,1]$ in the central partition. The partial rates and distortions for these codecells were computed in Example 2. We therefore read the partial rates and distortion off of the graph in Fig. 2, giving $(0.5, 12.5)$ and $(0.5, 0)$, respectively. The resulting Lagrangian cost is the edge weight for the graph in Fig. 7. Precisely, the arc (v_3, v_5) has weight $w_{3,5} = \nu_0 d((0,1]) + \nu_2 d((0,2]) + \lambda_2 r((0,2]) = 3 \cdot 0 + 1 \cdot 12.5 + 30 \cdot 0.5 = 27.5$. (Recall that the partial rate generated by the central codecell does not come into play.)

We next run the shortest path algorithm. In step ℓ, the algorithm finds the shortest path from node v_0 to node v_ℓ. The candidate solutions are all of the form: "shortest path to node v_k" (for some $1 \leq k < \ell$ for which there exists an edge (v_k, v_ℓ))

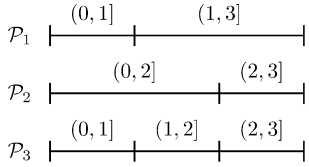

Fig. 8. Partitions created by the 2DSQ designed in Example 3.

followed by the edge (v_k, v_ℓ). The ordering rule guarantees that the shortest path to v_k is known before it is needed for the calculation of the shortest path to v_ℓ.

For our example, the first vertex in the sorted list is the origin $v_0 = (0,0)$. The shortest path from v_0 to v_0 has weight $s[0] = 0$. When $\ell = 1$, we seek the shortest path from v_0 to v_1. There is only one index $(k = 0)$ for which there is an arc (v_0, v_1). Thus, the shortest path from v_0 to v_1 comprises the shortest path from node v_0 to node v_0 (weight $s[0] = 0$) followed by the edge from v_0 to v_1 (weight $w_{0,1} = 0$). Thus, the shortest path from v_0 to v_1 has weight $s[1] = 0$. For $\ell \in \{2,3,4,5,6\}$, the process is similar; in each case, v_ℓ has only one incoming edge. Thus, the shortest path from v_0 to v_ℓ equals the shortest path from v_0 to that single predecessor followed by the edge from that predecessor to v_ℓ. The resulting weights are $s[2] = 0.5, s[3] = 0.5, s[4] = 169.25, s[5] = 28, s[6] = 15.5$. The case for $\ell = 7$ requires the first nontrivial calculation. Node v_7 has two incoming edges originating at v_1 and v_6. Thus, the shortest path to v_7 is either the shortest path to v_1 followed by the edge (v_1, v_7) or the shortest path to v_6 followed by the edge (v_6, v_7). The resulting weights are $s[1] + w_{1,7} = 0 + 15 = 15$ and $s[6] + w_{6,7} = 15.5 + 0.31 = 15.81$. Thus, the shortest path to node v_7 is $v_0 \to v_1 \to v_7$. The algorithm proceeds in the same manner for each vertex in turn. The final result is the shortest path from v_0 to v_{12} which describes the optimal 2DSQ partitions. The weight of this path is $s[12] = 43.31$ and the shortest path is $v_0 \to v_3 \to v_5 \to v_{11} \to v_{12}$. Recall that $v_0 = (0,0), v_3 = (1,0), v_5 = (1,2), v_{11} = (3,2), v_{12} = (3,3)$. Since edges that travel from top to bottom in the graph describe partition 1 codecells while edges that travel from left to right describe partition 2 codecells, this shortest path corresponds to threshold sequence $\{0,1,3\}$ for partition 1 and threshold sequence $\{0,2,3\}$ for partition 2—giving side partitions $\mathcal{P}_1 = \{C_{(0,1]}, C_{(1,3]}\}$ and $\mathcal{P}_2 = \{C_{(0,2]}, C_{(2,3]}\}$. The resulting central partition is $\mathcal{P}_0 = \{C_{(0,1]}, C_{(1,2]}, C_{(2,3]}\}$. These partitions are shown in Fig. 8. The resulting code achieves entropies 0.81 and 1.0 bit per symbol and expected squared-error distortions 66.67 and 12.5 in descriptions 1 and 2, respectively. The distortion of the central partition is 0 since when both descriptions are received there is no uncertainty about the symbol transmitted. □

Proof of Correctness

As in Section III, we must first exhibit a bijection between paths from $(0,0)$ to $(\hat{N}, \hat{N})$ and partition pairs on the alphabet $\{1, \ldots, \hat{N}\}$ and then show that the weight of a path equals the Lagrangian performance of the corresponding pair of partitions. Note that since both final partitions have the same alphabet, the lag is zero, and all partition pairs are valid.

For the bijection, one direction is immediate: each path describes how to "grow" a partition pair by successively extending one side partition at a time. We map each arc to the corresponding codecell being added to either $\mathcal{P}_1$ or $\mathcal{P}_2$; the codecell for an arc is chosen based on the labels of the origin and destination vertices of the arc.

Showing that for each $(\hat{N}, \hat{N})$ partition pair there is a unique path in the graph is a little more involved. First some terminology: given two partitions $\mathcal{P} = \{C_{(t_{i-1}, t_i]}\}_{i=1}^{l}$, $\mathcal{P}' = \{C_{(t'_{i-1}, t'_i]}\}_{i=1}^{l'}$ we say that $\mathcal{P}'$ is a *truncated version* of $\mathcal{P}$ if $l' \leq l$ and $t_i = t'_i$ for $i \in \{0, \ldots, l'\}$. Similarly, for two partition pairs $(\mathcal{P}_1, \mathcal{P}_2)$ and $(\mathcal{P}'_1, \mathcal{P}'_2)$, we define $(\mathcal{P}'_1, \mathcal{P}'_2)$ to be a truncated version of $(\mathcal{P}_1, \mathcal{P}_2)$ if $\mathcal{P}'_1$ and $\mathcal{P}'_2$ are truncated versions of $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively. (Since the partitions in each pair are presumed to be aligned on the left side, it follows that all four partitions are aligned on the left side.) We also say that some codecell C_i added to $\mathcal{P}'_i$ $(i \in \{1,2\})$ extends $(\mathcal{P}'_1, \mathcal{P}'_2)$ *according to* $(\mathcal{P}_1, \mathcal{P}_2)$ if the extension is still a truncated version of $(\mathcal{P}_1, \mathcal{P}_2)$. Note that if $(\mathcal{P}'_1, \mathcal{P}'_2)$ is not a truncated version of $(\mathcal{P}_1, \mathcal{P}_2)$, none of its extensions are either, regardless of how many codecells are added, since a mismatch in the threshold sequence cannot be repaired by adding more elements.

Now consider an arbitrary valid partition pair $(\mathcal{P}_1, \mathcal{P}_2)$ and let L be the sum of the number of codecells in $\mathcal{P}_1$ and the number of codecells in $\mathcal{P}_2$. We next show by induction that for any $l \in \{0, \ldots, L\}$ the graph contains exactly one length-l path that starts at $(0,0)$ and describes a truncated version of $(\mathcal{P}_1, \mathcal{P}_2)$. For $l = 0$, this is immediate. For length $l > 0$, any path of length l can be decomposed into a path of length $l-1$ and an extra arc. If the length-l path is to correspond to a truncated version of $(\mathcal{P}_1, \mathcal{P}_2)$, we have seen that the length-$l-1$ path must also. There exists only one path of length-$(l-1)$ that satisfies this condition, and the extension rule guarantees that there exists a unique arc that extends it according to $(\mathcal{P}_1, \mathcal{P}_2)$. (The latter statement uses the fact that both $(\mathcal{P}_1, \mathcal{P}_2)$ and the partition pair corresponding to the length-$(l-1)$ path are valid.) Therefore, the set of paths of length l satisfying the condition has cardinality exactly one. Note that our language implicitly uses the one-way correspondence from paths to partitions (which has already been demonstrated) but not its inverse.

The equivalence between path weight and Lagrangian performance is immediate by construction of the partial RD graph.

Extension From 2DSQ to MDSQ

An MDSQ with M descriptions may be described by M side partitions which induce $2^M - M - 1$ nontrivial *intersection partitions* (the analogues of the central partition from 2DSQ). In this section, we assume that the packet-loss scenarios corresponding to all of these intersection partitions occur with nonzero probability and thus that all are constrained; that assumption is relaxed in Section V. We describe each possible packet-loss scenario by a binary vector of length M (called the *packet-loss vector* and denoted by $\sigma \in \mathcal{B}^M$, where $\mathcal{B} = \{0,1\}$); in this vector, bit i is 1 if packet i is received and 0 if packet i is lost in the corresponding scenario. (For example, packet-loss vector $\sigma = (1101)$ describes the scenario where, of an initial four packets, all but packet three are received.) Side and intersection partitions can be indexed according to the same scheme (e.g.,

$\mathcal{P}_{(0010)}$ is the side partition used to describe packet three out of four, while $\mathcal{P}_{(1101)}$ is the intersection partition corresponding to combined knowledge of packets one, two, and four).

In this case, the achievable rate–distortion points are

$$\mathcal{D}^{\text{vr}} = \left\{ \left(R(\mathcal{P}_{0^{i-1}10^{M-i}})|_{i=1}^{M}, D(\mathcal{P}_\sigma)|_{\sigma \in \mathcal{B}^M} \right) \right\}$$

and M side-partitions that together achieve a point on the lower convex hull of $\mathcal{D}^{\text{vr}}$ minimize the Lagrangian

$$J^{\text{vr}}(\underline{\mathcal{P}}, \underline{\nu}, \underline{\lambda}) = \sum_{\sigma \in \mathcal{B}^M} \nu_\sigma D(\mathcal{P}_\sigma) + \sum_{i=1}^{M} \lambda_i R(\mathcal{P}_{0^{i-1}10^{M-i}})$$

for some nonnegative distortion and rate multipliers $\underline{\nu}$ and $\underline{\lambda}$ of dimensions 2^M (for the 2^M packet-loss scenarios, here indexed by $\sigma \in \mathcal{B}^M$) and M (for the M description rates), respectively. (For trivial intersection partition $\mathcal{P}_{(0\ldots0)}$, $D(\mathcal{P}_{(0\ldots0)})$ is independent of the side partitions and $\nu_{(0\ldots0)}$ is arbitrary.) We use $J^{\text{vr}}(\underline{\mathcal{P}}, \underline{\nu}, \underline{\lambda})$ as the optimization criterion for variable-rate MDSQ design.

To generalize our graph construction, we must generalize our partition extension rule and arc weight definitions. The partition extension rule generalization keeps the maximum lag between any two partitions less than or equal to 1 and fixes an order for breaking ties; we extend the shortest side partition, and if there are ties, we pick the competitor with the smallest index. Extending exactly one partition at a time according to the above extension rule implies that one side codecell is added and at most $2^{M-1} - 1$ *intersection codecells* are generated, resulting in at most 2^{M-1} distortion terms and one rate term in each arc weight.

More specifically, consider an arc (u, v) from vertex $u = (u_1, u_2, \ldots, u_M)$ to vertex $v = (v_1, v_2, \ldots, v_M)$. The extension rule implies that u and v may differ in only one component ($u_i = v_i$ for $i \in \{1, \ldots, j-1\}, \{j+1, \ldots, M\}$). Furthermore, u_j must be the shortest coordinate of vertex u (up to a possible tie), giving $u_j \leq u_i$ for all i. If there are ties ($u_j = u_l = v_l$ for some $l \neq j$), then necessarily $j < l$. In either case, side codecell $C_{(u_j, v_j]}$ is added (at a Lagrangian cost of $\lambda_j r(C) + \nu_{(0^{j-1}10^{M-j})} d(C)$). Moreover, for each packet loss scenario $\sigma \in \mathcal{B}^M$, if $\sigma_j = 1$, then intersection codecell $C_{(u_j, v_\star]}$, where $v_\star = \min\{v_i : \sigma_i = 1\}$, is generated in intersection partition $\mathcal{P}_\sigma$ (at cost $\nu_\sigma d(C)$); otherwise, no codecells are generated in $\mathcal{P}_\sigma$. Adding up the Lagrangian costs from the side codecell and any possible intersection codecells gives the weight for arc (u, v); note that only the side codecell contributes rate to the total cost (thus, there is a single rate term in the arc weight).

Some of the $(\hat{N}+1)^M$ vertices are inaccessible due to the extension rule. Consider vertex v. If $v_i = 0$ for some i, then either $v_j = 0$ for all $j > i$, or else the vertex is inaccessible since access to that vertex would require extending partition j before extending partition i, even though $j > i$ (contradicting the extension rule). It turns out that this necessary condition is also sufficient for accessibility. Any accessible vertex $v = (v_1, v_2, \ldots, v_j, 0, \ldots, 0)$ can be obtained in j steps starting from $(0, \ldots, 0)$: at step i, extend side partition i from 0 to v_i, for i from 1 to j.

In the course of the shortest path algorithm, one of the required steps is to obtain the ancestor set of a vertex v. Assume that side partition j has been extended. Since $u_j < v_j$, and $u_j = \min\{u_i\}$, we get $u_j \leq \min\{v_i\}$. If there are no ties, the inequality must be strict. If there are ties, then one of the tiers, namely, u_j, has been extended, but the other tiers are unchanged, so $u_j = \min\{v_i\}$ and u_j must be the tier with the least index in u. In summary, if v_l is the shortest component of v (if there are ties, choose the one with the least index, i.e., minimize l). Then v has $Mv_l + l - 1$ ancestors: Mv_l "untied" ancestors ($j \in \{1, \ldots, M\}$ and $u_j \in \{0, \ldots, v_l - 1\}$) and $l - 1$ "tied" ancestors ($j \in \{1, \ldots, l-1\}$ and $u_j = v_l$). When $v_l = 0$ all ancestors are "tied."

The size of the ancestor set of a given vertex equals its in-degree; therefore, the number determined above can be plugged directly into the topological sorting algorithm. A slight simplification is possible. If we eliminate inaccessible vertices from the graph, then the origin is the only remaining vertex with in-degree zero, making initialization trivial for both the topological sorting and the dynamic programming stages of the shortest path algorithm. We also need to do away with inaccessible ancestors in our account. If $v_l > 0$, this means that $u_j = 0$ is permitted only for $j = M - 1$, thus leaving $M(v_l - 1) + l$ accessible ancestors; if $v_l = 0$ but $l > 1$, then only one ancestor is accessible, with $j = l - 1$ and $u_j = 0$; finally if $l = 1$ and $v_l = 0$ then v is the origin, which has no ancestors.

Complexity

The computational complexity of the algorithm depends on the size of the vertex set and edge set of the partial RD graph. With M descriptions, there are $(\hat{N}+1)^M$ vertices; each vertex has at most $\hat{N}$ outgoing arcs, thus there are $O((\hat{N}+1)^{M+1})$ edges. The complexity of computing the arc weights dominates all the other steps of the code design process. In general, up to 2^{M-1} codecells are generated for each edge, and we must calculate the boundaries for those codecells. Each boundary calculation involves taking the intersection of some subset of the M side partitions. Computing the intersection of partitions $\mathcal{P}_i$ and $\mathcal{P}_j$ requires only $O(|\mathcal{P}_i| + |\mathcal{P}_j|)$ operations since most pairs of codecells do not intersect. Specifically, we can find the intersection partition $\mathcal{P}$ using a merge–sort on the threshold sequences of the side partitions. That is, if $\mathcal{P}_i = \{(t_0, t_1], \ldots, (t_{K-1}, t_K]\}$ and $\mathcal{P}_j = \{(u_0, u_1], \ldots, (u_{L-1}, u_L]\}$, then $\mathcal{P}$ is the partition corresponding to the merge sort of thresholds $\{t_0, \ldots, t_K\}$ and $\{u_0, \ldots, u_L\}$. In general, computing the boundary of a codecell requires examining at most M coordinates; the algorithm runs in time $O(M(2(\hat{N}+1))^{M+1})$.

Fixed-Rate MDSQ

In fixed-rate MDSQ design, the side-partition sizes are fixed and the rate of each description equals the logarithm of the corresponding side-partition size. Let d_k be the fixed size of partition $\mathcal{P}_{0^{k-1}10^{M-k}}$. Then

$$\mathcal{D}^{\text{fr}}(\underline{d}) = \{(D(\mathcal{P}_\sigma)|_{\sigma \in \mathcal{B}^M}) : |\mathcal{P}_{0^{k-1}10^{M-k}}| \leq d_k \ \forall k\}$$

describes the set of achievable distortion points at fixed rate $(\log d_1, \ldots, \log d_M)$. In this case, M side-partitions achieving

a point on the lower convex hull of $\mathcal{D}^{\text{fr}}$ minimize the Lagrangian performance measure

$$J^{\text{fr}}(\underline{\mathcal{P}}, \underline{d}, \underline{\nu}) = \sum_{\sigma \in \mathcal{B}^M} \nu_\sigma D(\mathcal{P}_\sigma)$$

for some vector of nonnegative Lagrangian parameters $\underline{\nu}$. We use $J^{\text{fr}}(\underline{\mathcal{P}}, \underline{\nu})$ as the optimization criterion for fixed-rate MDSQ design.

The problem of fixed-rate MDSQ design differs from that of variable-rate MDSQ design both in the optimization criterion and in the constraint on the size of each side partition. While the first difference affects only the edge weights of the partial RD graph, the second difference affects the node definitions. Let node $((v_1, s_1), \ldots, (v_M, s_M))$ represent the set of all possible collections of M side partitions $(\mathcal{P}_1, \ldots, \mathcal{P}_M)$ such that $\mathcal{P}_i$ spans $(0, v_i]$ with $|\mathcal{P}_i| = s_i$ segments. The partition extension rule is generalized accordingly, again keeping the lag between any two side partitions to at most 1 and here updating the partition size parameters appropriately.

Specifically, we have $(\hat{N}+1)^M \prod_{i=1}^M (d_i+1)$ vertices $\{((v_1, s_1), \ldots, (v_M, s_M))\}_{v_i \in \{0,\ldots,\hat{N}\}, s_i \in \{0,\ldots,d_i\}}$. (The extension rule makes some of the vertices inaccessible.) For any arc (u, v) from vertex $u = ((u_1, r_1), \ldots, (u_M, r_M))$ to vertex $v = ((v_1, s_1), \ldots, (v_M, s_M))$, the extension rule again implies that $u_i = v_i$ and $r_i = s_i$ for all $i \neq j$, $u_j \leq u_i$ for all i, and $j < l$ for any l such that $u_j = u_l = v_l$. Further, since arc (u, v) corresponds to the addition of a single side codecell to the jth partition, $s_j = r_j + 1$. For any $\sigma \in \mathcal{B}^M$ with $\sigma_j = 1$, arc (u, v) implies the addition of codecell $C_{(u_j, v_\star]}$ $(v_\star = \min\{v_i : \sigma_i = 1\})$ to partition $\mathcal{P}_\sigma$ at Lagrangian cost $\nu_\sigma d(C_{(u_j, v_\star]})$. Adding these Lagrangian costs gives the weight for arc (u, v). Fixed-rate MDSQ optimization is equivalent to finding the shortest path from $((0,0), \ldots, (0,0))$ to $((\hat{N}, d_1), \ldots, (\hat{N}, d_M))$. The complexity of the resulting algorithm is $O(M(2(\hat{N}+1))^{M+1} \prod_{i=1}^M (d_i+1))$.

V. MRSQ and Other Restricted MDSQs

In this section, we replace the MDSQ assumption that any packet-loss scenario $\sigma \in \mathcal{B}^M$ occurs with nonzero probability by the restricted MDSQ assumption that only σ in some *admissible scenario set* $\Sigma \subseteq \mathcal{B}^M$ occur with nonzero probability. Under this new assumption, the optimal rate for describing a side partition with fixed- and variable-rate codes varies with Σ. We begin by studying multiresolution codes $(\Sigma = \{(1^i 0^{M-i}) : i \in \{1, \ldots, M\}\})$ and then generalize to a broader class of admissible scenario sets.

Again, we focus initially on variable-rate codes, leaving the discussion of fixed-rate coding to the end of the section. The algorithms that follow again constrain every codecell of every side partition to be convex, which creates convex codecells in all intersection partitions as well. Any MRSQ with convex codecells at all resolutions can be represented with a collection of partitions that satisfies these constraints.

MRSQ Overview

In MRSQ, packet i is never received without packets $\{1, \ldots, i-1\}$, giving $\Sigma = \{(1^i 0^{M-i}) : i \in \{1, \ldots, M\}\}$.

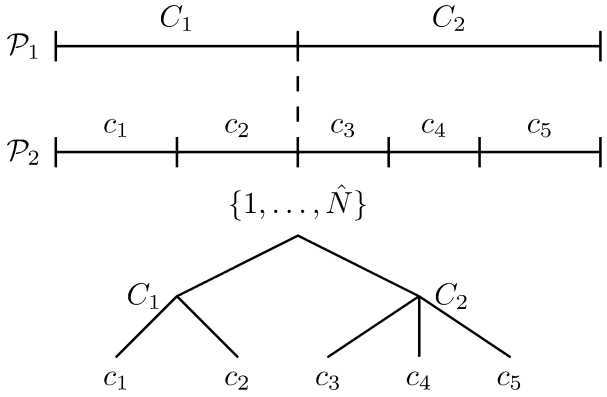

Fig. 9. Successive refinement of partitions in an MRSQ. In this example, $\mathcal{P}_2(C_1) = \{c_1, c_2\}$ and $\mathcal{P}_2(C_2) = \{c_3, c_4, c_5\}$. The encoder describes an M-step path from the root to a leaf in the tree structure, uniquely describing the representations associated with all M resolutions.

Given this constraint, for any $i > 1$, packet i need not be uniquely decodable on its own. Instead, it suffices for packet i to be *conditionally* uniquely decodable given packets $1, \ldots, i-1$.

Let $\mathcal{P}_i = \mathcal{P}_{(1^i 0^{M-i})}$ for each $i \in \{1, \ldots, M\}$. We begin by examining the partitions $\{\mathcal{P}_i\}_{i=1}^M$. By definition of the intersection partitions, for each $C \in \mathcal{P}_i$, $C = \cap_{j=1}^i C_j$ for some $C_j \in \mathcal{P}_{\sigma(j)}, j \in \{1, \ldots, i\}$. Thus, for each $i \in \{1, \ldots, M\}$, partition $\mathcal{P}_i$ is a *refinement* of $\mathcal{P}_{i-1}$; that is, the threshold sequence of $\mathcal{P}_{i-1}$ is a subsequence of the threshold sequence of $\mathcal{P}_i$. (Here $\mathcal{P}_0 = \{1, \ldots, \hat{N}\}$ by definition.) Since $\mathcal{P}_i$ refines $\mathcal{P}_{i-1}$ (written $\mathcal{P}_i \succeq \mathcal{P}_{i-1}$), for any $C \in \mathcal{P}_{i-1}$ there exists a collection of cells $c_1, \ldots, c_k \in \mathcal{P}_i$ so that $\mathcal{P}_i(C) = \{c_1, \ldots, c_k\}$ is a partition of C. Fig. 9 shows the refinement relationship between partitions $\mathcal{P}_1$ and $\mathcal{P}_2$.

The restriction that $\{r(C) : C \in \mathcal{P}_i\}$ satisfies Kraft's inequality for each $i \in \{1, \ldots, M\}$ is necessary in MDSQ since it allows the decoder to uniquely decode description i when that is the only description received. This restriction is not necessary for MRSQ. For any $i > 1$, the decoder uses the first $i-1$ descriptions to determine some codecell $C \in \mathcal{P}_{i-1}$; since $\mathcal{P}_i$ refines $\mathcal{P}_{i-1}$, the ith incremental description need only distinguish between the members of $\mathcal{P}_i(C)$ rather than all of the members of $\mathcal{P}_i$. Thus unique decodability in an MRSQ is achieved if and only if $\{r(c) : c \in \mathcal{P}_i(C)\}$ satisfies Kraft's inequality for each $i \in \{1, \ldots, M\}$ and $C \in \mathcal{P}_{i-1}$. The optimal rate for describing cell $c \in \mathcal{P}_i(C)$ given a prior description of the cell $C \in \mathcal{P}_{i-1}$ that it refines is $-\log p(c|C)$, giving optimal partial rate $r(c|C) = -p(c|C) \log p(c|C)$, where $p(c|C) = p(c)/p(C)$. The expected ith-resolution rate is

$$R(\mathcal{P}_i|\mathcal{P}_{i-1}) = -\sum_{C \in \mathcal{P}_{i-1}} p(C) \sum_{c \in \mathcal{P}_i(C)} p(c|C) \log p(c|C).$$

This *conditional* entropy bound on the expected rate is approximated in practice by conditional entropy codes.

Optimization Criterion

The rate-distortion points achievable by variable-rate MRSQs are

$$\mathcal{D}^{\text{vr}} = \{(R(\mathcal{P}_i|\mathcal{P}_{i-1}), D(\mathcal{P}_i))|_{i=1}^M : \mathcal{P}_M \succeq \cdots \succeq \mathcal{P}_0\}$$

and M successively refined partitions that together achieve a point on the lower convex hull of $\mathcal{D}^{\text{vr}}$ minimize the Lagrangian

$$J^{\text{vr}}(\underline{\mathcal{P}}, \underline{\nu}, \underline{\lambda}) = \sum_{i=1}^{M} [\nu_i D(\mathcal{P}_i) + \lambda_i R(\mathcal{P}_i|\mathcal{P}_{i-1})] \qquad (2)$$

for some positive vectors $\underline{\nu} = \{\nu_i\}_{i=1}^{M}$ and $\underline{\lambda} = \{\lambda_i\}_{i=1}^{M}$. We therefore use $J^{\text{vr}}(\underline{\mathcal{P}}, \underline{\nu}, \underline{\lambda})$ as our design criterion. Note that

$$\begin{aligned}
&J^{\text{vr}}(\underline{\mathcal{P}}, \underline{\nu}, \underline{\lambda}) \\
&\quad \overset{\text{(a)}}{=} \sum_{i=1}^{M} \sum_{C \in \mathcal{P}_{i-1}} \sum_{c \in \mathcal{P}_i(C)} [\nu_i d(c) + \lambda_i p(C) r(c|C)] \\
&\quad \overset{\text{(b)}}{=} \sum_{i=1}^{M} \sum_{C \in \mathcal{P}_{i-1}} \sum_{c \in \mathcal{P}_i(C)} \left[\nu_i d(c) + \lambda_i p(c) \log \frac{p(C)}{p(c)}\right] \\
&\quad \overset{\text{(c)}}{=} \sum_{i=1}^{M} \sum_{C \in \mathcal{P}_{i-1}} \left[-\lambda_i r(C) + \sum_{c \in \mathcal{P}_i(C)} [\nu_i d(c) + \lambda_i r(c)]\right] \\
&\quad \overset{\text{(d)}}{=} \sum_{i=1}^{M} \sum_{C \in \mathcal{P}_{i-1}} \sum_{c \in \mathcal{P}_i(C)} [\nu_i d(c) + (\lambda_i - \lambda_{i+1}) r(c)],
\end{aligned}$$

where (a) and (b) expand the terms in (2); (c) follows from $\sum_{c \in \mathcal{P}_i(C)} p(c) = p(C)$; and (d) uses the definition $\lambda_{M+1} = 0$ and the fact that $\mathcal{P}_0$ has only one cell C with $r(C) = 0$. We express the design criterion recursively as

$$J^{(i)}(C) = \min_{\mathcal{P}_i(C)} \sum_{c \in \mathcal{P}_i(C)} \left[\nu_i d(c) + (\lambda_i - \lambda_{i+1}) r(c) + J^{(i+1)}(c)\right] \qquad (3)$$

with $J^{(M+1)}(c) = 0$ for all c and

$$J^{\text{vr}}(\underline{\mathcal{P}}, \underline{\nu}, \underline{\lambda}) = J^{(1)}(\{1, \ldots, \hat{N}\}).$$

Algorithm

The design algorithm finds the set of successively refined partitions $\underline{\mathcal{P}}$ (compatible with the coarse grid $\mathcal{G}$) that minimizes $J^{(1)}(\{1, \ldots, \hat{N}\})$ for a fixed collection of Lagrangian parameters $\underline{\lambda}$ and $\underline{\nu}$. Notice that for $J^{(i)}(C)$ to be optimal, $J^{(i+1)}(c)$ must be optimal for each $c \in \mathcal{P}_i(C)$; each of the $J^{(i+1)}(c)$ terms in the equation for $J^{(i)}(C)$ can be optimized independently of the others since the cells c in $\mathcal{P}_i(C)$ do not overlap. These observations suggest a bottom-up dynamic programming solution. For each possible codecell C, we find the partition $\mathcal{P}_M(C)$ of C that minimizes

$$J^{(M)}(C) = \sum_{c \in \mathcal{P}_M(C)} [\nu_M d(c) + (\lambda_M - \lambda_{M+1}) r(c)]$$

using a shortest path algorithm. Given $J^{(M)}(\cdot)$ for *all* $\mathcal{G}$-compatible codecells, we similarly find $\mathcal{P}_{M-1}(C)$ for each C by minimizing

$$J^{(M-1)}(C) = \sum_{c \in \mathcal{P}_{M-1}(C)} \left[\nu_{M-1} d(c) + (\lambda_{M-1} - \lambda_M) r(c) + J^{(M)}(c)\right].$$

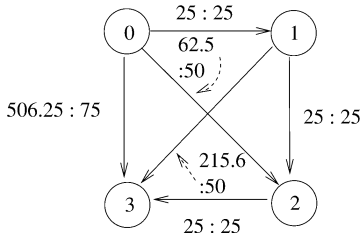

Fig. 10. Weights and resulting Lagrangian costs for resolution 2 in Example 4. Edge (v, v') has label $w_2(v, v') : J^{(2)}(v, v')$.

Iterating back from M to 1 yields

$$J^{(1)}(\{1, \ldots, \hat{N}\}) = J^{\text{vr}}(\underline{\mathcal{P}}, \underline{\nu}, \underline{\lambda}).$$

More formally, we begin by initializing $J^{(M+1)}(C) = 0$ for all $C = C_{(a,b]}$ with $a, b, \in \{0, \ldots, \hat{N}\}$. We then proceed through M iterations, indexed from M to 1. Each iteration relies on a different partial RD graph. In iteration i, for all $\mathcal{G}$-compatible codecells C we compute and store the subpartition $\mathcal{P}_i(C)$ that minimizes $J^{(i)}(C)$. The optimal $J^{(i)}(C)$ is found by running the shortest path algorithm between each pair of distinct vertices in a graph with weight

$$\begin{aligned}
&w_i(v, v') \\
&\quad = \nu_i d(C_{(v,v']}) + (\lambda_i - \lambda_{i+1}) r(C_{(v,v']}) + J^{(i+1)}\left(C_{(v,v']}\right)
\end{aligned}$$

on the arc from node v to node $v'(v, v' \in \{0, \ldots, \hat{N}\})$. Here $J^{(i+1)}(C_{(v,v']})$ is the weight on the shortest path from node v to node v' using arc weights $\{w_{i+1}(a, b)\}_{a,b=0}^{\hat{N}}$; that shortest path is known from the previous iteration. (We implicitly use the two-way correspondence between arc (v, v') and codecell $C_{(v,v']}$.) The final results of the algorithm are $J^{(1)}(\{1, \hat{N}\})$ and the successively refined partitions $\{\mathcal{P}_1, \ldots, \mathcal{P}_M\}$ that achieve it. To construct the partitions, we iterate back from 1 to M. Partition $\mathcal{P}_i$ is constructed from $\mathcal{P}_{i-1}$ by pasting together $\mathcal{P}_i(C)$ for all $C \in \mathcal{P}_{i-1}$.

For each resolution, we find the shortest paths from node v to node v' for all $v, v' \in \{1, \ldots, \hat{N}\}$ simultaneously using an *all-pairs shortest path* algorithm (see [20]) rather than calculating the shortest path between each pair separately. This reduces the complexity from $O((\hat{N}+1)^4)$ per resolution, to $O((\hat{N}+1)^3)$ per resolution. Thus, the overall complexity is $O(M(\hat{N}+1)^3)$.

Example 4: We design a 2RSQ using the pmf from Example 2. Recall that Fig. 2 describes the partial rates and distortions for this example. Our Lagrangian multipliers are $\lambda_1 = 100$ and $\nu_1 = 1$ for the first resolution and $\lambda_2 = 50$ and $\nu_2 = 3$ for the second resolution.

The algorithm begins by finding all-pair shortest paths for resolution 2, in a graph with weights $w_2(v, v') = \nu_2 d(v, v') + \lambda_2 r(v, v')$. Fig. 10 shows the weight $w_2(v, v')$ of edge (v, v') and weight $J^{(2)}(C_{(v,v']})$ of the shortest path from v to v'. The following table shows a shortest path from node v to node v'. In this example, it turns out that each shortest path uses the maximum number of possible edges. For example, the shortest path from 0 to 2 comprises the edge from 0 to 1 (weight 25) followed by the edge from node 1 to node 2 (weight 25) to give

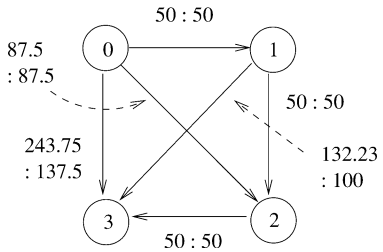

Fig. 11. Weights and resulting Lagrangian costs for resolution–1 in Example 4. Edge (v, v') has label $w_1(v, v') : J^{(1)}(v, v')$.

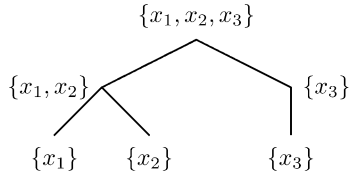

Fig. 12. The 2-resolution tree corresponding to the code designed in Example 4.

a total weight of 50. This is the shortest path because the alternative—a single edge, passing directly from node 0 to node 2 without stopping at node 1—has a higher Lagrangian weight, 62.5.

		v'		
		1	2	3
v	0	$0 \to 1$	$0 \to 1 \to 2$	$0 \to 1 \to 2 \to 3$
	1		$1 \to 2$	$1 \to 2 \to 3$
	2			$2 \to 3$

For resolution 1, the all-pairs shortest path algorithm runs on a graph with weights $w_1(v, v') = \nu_1 d(v, v') + (\lambda_1 - \lambda_2) r(i, j) + J^{(2)}(v, v')$. Fig. 4 shows these weights and the resulting shortest path weights $J^{(1)}(v, v')$. The following table describes the corresponding shortest paths.

		v'		
		1	2	3
v	0	$0 \to 1$	$0 \to 2$	$0 \to 2 \to 3$
	1		$1 \to 2$	$1 \to 2 \to 3$
	2			$2 \to 3$

For instance, let us again find the shortest path between nodes 0 and 2, this time at resolution 1. The relevant weights are $w_1(0,1) = w_1(1,2) = 0 + 50 \cdot 0.5 + 25 = 50$ and $w_1(0,2) = 12.5 + 50 \cdot 0.5 + 50 = 87.5$. Therefore, the shortest path is the direct edge $(0, 2)$ with Lagrangian cost 87.5. (The path $0 \to 1 \to 2$ has a larger total cost of 100.)

The optimal 2RSQ with convex codecells is described by the shortest path from 0 to 3 in the resolution 1 graph (see Fig. 11). This path has Lagrangian weight 137.5 and is achieved by path $0 \to 2 \to 3$, which corresponds to resolution-1 codecells $C_{(0,2]} = \{x_1, x_2\}$ and $C_{(2,3]} = \{x_3\}$. Resolution-1 codecell $C_{(v,v']}$ is partitioned in resolution 2 by the codecells corresponding to the shortest path from node v to node v' in the resolution-2 graph. Thus, $C_{(0,2]}$ is partitioned by $C_{(0,1]}, C_{(1,2]}$ in resolution 2 while $C_{(2,3]}$ is partitioned by $C_{(2,3]}$ in resolution 2. Figs. 12 and 13 show the corresponding MRSQ tree and partitions. □

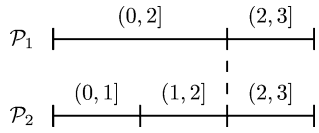

Fig. 13. Optimal partitions for the 2RSQ in Example 4.

Restricted MDSQ

We next modify the general MDSQ design algorithm for use with more general scenario sets. The modifications greatly increase computational complexity, but the algorithm remains polynomial in the size of the source alphabet. Since there exist more efficient specialized algorithms for some restricted-MDSQ scenarios (e.g., the above bottom-up dynamic programming algorithm for MRSQ design), the generalization is not of interest for all values of Σ. It does, however, increase the collection of scenario sets for which we can find the best code among all codes that meet the convexity constraint.

In MRSQ, packet i never arrives without packets $\{1, \ldots, i-1\}$, so we can save rate by allowing packet i's description to depend on the descriptions of prior packets. The *dependency sets* $\{\delta[i]\}_{i=1}^{M}$ capture a similar property for general restricted MDSQs. Packet i *depends* on packet j ($j \in \delta[i]$) if i is never received without j (i.e., $\sigma_i = 1$ implies $\sigma_j = 1$ for all admissible scenarios $\sigma \in \Sigma$). We call any $j \in \delta[i]$ a *requisite* of i, and we say that packet i is *independent* if $\delta[i] = \phi$. For example, in a 3RSQ (three-resolution scalar quantizer) $\Sigma = \{(100), (110), (111)\}$, $\delta[1] = \phi, \delta[2] = \{1\}$, and $\delta[3] = \{1, 2\}$, packet 1 is independent, packet 1 is a requisite of packet 2, and packets 1 and 2 are requisites of packet 3.

In MRSQ, if packet i depends on j, then partition $\mathcal{P}_i$ must be a refinement of $\mathcal{P}_j$, and the optimal rates are the corresponding conditional entropies. Similar refinement relationships and conditional lossless codes also arise in other restricted MDSQs, and the algorithm that follows uses the dependency sets $\{\delta[i]\}_{i=1}^{M}$ to determine these relationships.

While knowing the dependency sets is sufficient for finding the optimal restricted MDSQ for many scenario sets, dependency sets capture only the simplest type of dependence, and thus the algorithm discussed below does not guarantee the best convex-codecell code for all scenario sets. For example, consider the scenario set $\Sigma = \{(001), (010), (101), (110)\}$. Here packet 1 is guaranteed to appear with either packet 2 or packet 3. Yet $\delta[1] = \phi$, and thus the algorithm that follows treats packet 1 as an independent packet, calculating its rate as the full entropy of $\mathcal{P}_{\sigma(i)}$ Since this entropy is not necessarily optimal [31], the algorithm guarantees an optimal solution only for examples characterized entirely by the simple dependencies captured by δ. Examples of scenario sets covered here but not covered previously include $\Sigma = \{(100), (110), (001), (101), (111)\}$ (a 2RSQ plus an independent packet), $\Sigma = \{(100), (110), (101)\}$ (packets 2 and 3 never appear without 1), and $\Sigma = \{(0001), (0011), (0100), (0101), (0111), (1100), (1101), (1111)\}$ (a multiresolution multiple description code).

One way to compute the dependency sets is to check whether $j \in \delta[i]$ for all pairs $i, j \in \{1, \ldots, M\}$ with $i \neq j$ by examining

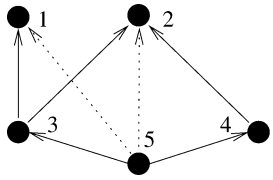

Fig. 14. The dependency graph for $\Sigma = \{10000, 01000, 11100, 01010, 11110, 11111\}$; dashed lines show indirect dependencies.

all $\sigma \in \Sigma$; this procedure takes $O(M^2|\Sigma|)$ operations, giving $O(M^2 2^M)$ in the worst case. There is not much room for improvement here since any algorithm must visit each admissible scenario at least once.

The dependency function δ implicitly defines a (directed) *dependency graph* with M vertices and an arc from i to j if i depends on j ($j \in \delta[i]$). An example appears in Fig. 14. (We remove the dotted arcs in the discussion that follows, but for now they are part of the graph). There are no restrictions on the in- and out-degrees of each vertex (i.e., one packet can depend on many others and multiple packets can depend on the same packet). For any two mutually dependent packets i and j, we replace i and j with a new "joint" packet k; since all packets are jointly encoded and i and j can always be jointly decoded, the rate needed to describe the new packet k can be broken up arbitrarily between the two original packets i and j.

We sort the vertices of the graph (the packets) topologically, renumbering them so that if i is dependent on j, then $i > j$. (Since all mutually dependent packets are merged, the graph is guaranteed to be acyclic.) The ordering is useful for the design stage, where we again use a tie-breaking rule that extends the competitor with the least index.

We calculate the rate of dependent packets using the corresponding conditional entropies. To calculate these conditional entropies, we enforce refinement relationships on some of the side partitions. In MRSQs, if packet i depends on j, then $\mathcal{P}_i$ must be a refinement of $\mathcal{P}_j$. In general restricted MDSQs, if packet i depends on j, then for each $\sigma, \sigma' \in \Sigma$ with $\sigma_i = \sigma_j = 1, \sigma'_k = \sigma_k$ ($k \neq i$), and $\sigma'_i = 0$, $\mathcal{P}_\sigma$ must be a refinement of $\mathcal{P}_{\sigma'}$. Unfortunately, this refinement relationship between intersection partitions implies no refinement relationships between side partitions. In fact, refinement relationships are not necessary in the side partitions of dependent packets. We next show, however, that considering only side partitions for which $j \in \delta[i]$ implies $\mathcal{P}_{\sigma(i)}$ refines $\mathcal{P}_{\sigma(j)}$ causes no loss of generality.

For any $\Sigma \subset \mathcal{B}^M$, the set of side partitions $\{\mathcal{P}_{\sigma(i)}\}_{i=1}^M$ that yields the full partition set $\{\mathcal{P}_\sigma\}_{\sigma\in\Sigma}$ is not necessarily unique. We next show that for any full partition set $\{\mathcal{P}'_\sigma\}_{\sigma\in\Sigma}$ derived from arbitrary side partition set $\{\mathcal{P}'_{\sigma(i)}\}_{i=1}^M$, there exists another side partition set $\{\mathcal{P}_{\sigma(i)}\}_{i=1}^M$ for which $\{\mathcal{P}_\sigma\}_{\sigma\in\Sigma} = \{\mathcal{P}'_\sigma\}_{\sigma\in\Sigma}$ and $j \in \delta[i]$ implies $\mathcal{P}_{\sigma(i)} \succeq \mathcal{P}_{\sigma(j)}$. We construct $\{\mathcal{P}_{\sigma(i)}\}_{i=1}^M$ as $\mathcal{P}_{\sigma(i)} = \mathcal{P}'_{\hat{\sigma}(i)}$, where $\hat{\sigma}_j(i) = 1$ for all $j \in \delta[i] \cup \{i\}$ and $\hat{\sigma}_j(i) = 0$ otherwise. Then $\{\mathcal{P}_\sigma\}_{\sigma\in\Sigma} = \{\mathcal{P}'_\sigma\}_{\sigma\in\Sigma}$ since packet i never appears without its requisite packets.

Since refinement is transitive ($\mathcal{P}_{\sigma(i)} \succeq \mathcal{P}_{\sigma(j)}$ and $\mathcal{P}_{\sigma(j)} \succeq \mathcal{P}_{\sigma(l)}$ implies $\mathcal{P}_{\sigma(i)} \succeq \mathcal{P}_{\sigma(l)}$), some of the refinement relationships contained in the initial dependency graph are superfluous. We say that j *depends directly* on i (and write $j \in \Delta[i]$) if and only if j depends on i ($j \in \delta[i]$) and j does not depend on any other node l that also depends on i (there does not exist an $l \in \delta[i]$ such that $j \in \delta[l]$). We remove all but the direct dependencies. From a graph-theoretical perspective, this operation is the inverse of the transitive closure problem on the dependency graph.

We are now ready to extend the MDSQ design algorithm. First, we modify the partition extension rule to ensure successive refinement according to Δ. Since the packets are sorted topologically, dependent packets have higher indices, and their partitions are extended only after those of their requisites. To preserve refinement, we make sure that cells added to dependent partitions never "cross" boundaries of cells already present in requisite partitions. Since codecell lag is always 1 or 0, the only place where such a crossing might occur is at the end of a requisite partition. Therefore, we remove from the partial RD graph any arc that extends a dependent partition beyond the end of any of its requisite partitions.

Since arc costs for codecells added to dependent partitions measure rate as conditional entropy, we must modify the partial RD graph to include information about the "parent" codecells in all requisite partitions. Given the bound on codecell lag, each "parent" is the last codecell in its respective partition. Thus, for each packet with dependents, we modify the vertex label to describe not only the end of the corresponding partition but also the beginning of the final codecell in that partition; thus, instead of keeping track of only the last element in the threshold sequence for that partition, we keep track of the last two elements. This *double labeling* increases the number of vertices from $(\hat{N}+1)^M$ to $O((\hat{N}+1)^{2M})$ in the worst case.

Each (single or double) partition label is a component in the M-dimensional label for a vertex; partition labels are elements of $\{0, \ldots, \hat{N}\} \cup \{0, \ldots, \hat{N}\}^2$, so vertex labels are elements of $(\{0, \ldots, \hat{N}\} \cup \{0, \ldots, \hat{N}\}^2)^M$. As an example, if $\delta[1] = \delta[3] = \phi$ and $\delta[2] = 1$ (packets 1 and 3 are independent, while packet 2 depends on 1), a vertex label might look like $((v_1, v'_1), v_2, v_3) = ((2, 5), 2, 7)$—corresponding to partitions $\mathcal{P}_{\sigma(1)}$ extending up to the fifth cell of the coarse alphabet, with its final codecell starting at the third cell, $\mathcal{P}_{\sigma(2)}$ extending up to the second cell of the coarse alphabet, and $\mathcal{P}_{\sigma(3)}$ up to the seventh cell of the coarse alphabet. As illustrated in this example, double labels (v_i, v'_i) consist of the last two elements in the threshold sequence of partition $\mathcal{P}_{\sigma(i)}$ (corresponding to a final codecell $C_{(v_i,v'_i]}$); an exceptional situation arises when the partition is empty (the threshold sequence has only one element) in which case we use the label $(0, 0)$.

All other partition extension rules remain valid once we realize that the second element in a double label denotes the end of a partition—in particular, this is how we measure the length of a partition when determining which partition in a tuple is the shortest. The introduction of double labels raises three new issues. First, the successor set of a vertex may involve extending a double-labeled partition, in which case the following rule applies: double-labeled partition $\mathcal{P}_{\sigma(i)}$ can be extended from (v_i, v'_i) to (v'_i, v''_i) where $v''_i > v'_i$ (provided all the other extension rules are obeyed). Second, the ancestor set of a vertex may involve "shortening" a double-labeled partition $\mathcal{P}_{\sigma(i)}$; $(v'_i.v''_i)$ can be shortened to (v_i, v'_i) where either $v_i < v'_i$ or $v_i = v'_i = 0$.

Finally, the weight on the arcs associated with dependent packets in the partial RD graph must be changed to use conditional entropy in the rate calculations (distortion terms are left unchanged). As noted in Section IV, only side codecells contribute rate terms to the Lagrangian cost. Given the double labels on requisite packets and the enforced refinement condition for dependent side partitions, for any packet i, any codecell $c \in \mathcal{P}_{\sigma(i)}$, and any requisite packet $j \in \delta[i]$, the "parent" codecell $C_j \in \mathcal{P}_{\sigma(j)}$ with $c \subseteq C_j$ is described by the double-label index for packet j. Since this parent codecell is known, we can calculate the optimal rate for describing packet c given descriptions of $\{C_j\}_{j\in\delta[i]}$ as

$$r(c\,|\,C_j, j \in \delta[i]) = -p(c\,|\,C_j, j \in \delta[i]) \log p(c\,|\,C_j, j \in \delta[i])$$

giving a total rate contribution for group c equal to

$$p(C_j, j\in\delta[i]) r(c\,|\,C_j, j\in\delta[i]) = -p(c) \log p(c\,|\,C_j, j\in\delta[i]).$$

Here

$$p(c\,|\,C_j, j \in \delta[i]) = p(c)/p(C_j, j \in \delta[i])$$

and

$$p(C_j, j \in \delta[i]) = p(\cap_{j\in\delta[i]} C_j).$$

Fixed-Rate Coding

The extension to fixed-rate code design proceeds in a manner similar to the one described in Section IV. Again, the side-partition sizes are fixed, the nodes of the partial RD graph are modified to include partition sizes, and the entropies and conditional entropies are replaced by the corresponding log-cardinalities.

VI. Decoder Side Information

We next consider the design of scalar quantizers with decoder side information. The proposed approach allows the incorporation of decoder side information in any of the above coding scenarios, yielding fixed- and variable-rate conventional scalar quantizers with decoder side information (Wyner–Ziv codes), MRSQs with decoder side information, and MDSQs with decoder side information. We begin by describing the design algorithm for ECSQ with decoder side information—giving side-information ECSQ (SECSQ). We then generalize to other code types. In each case, the algorithm finds the best code among all codes of the given type with convex codecells.

ECSQ With Decoder Side Information

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the ordered, scalar source and side-information alphabets, respectively. By assumption, $\mathcal{X} \times \mathcal{Y}$ is effectively finite, and the joint pmf on $\mathcal{X} \times \mathcal{Y}$ is known. Let the source- and side-information-alphabet sizes be $N(\mathcal{X})$ and $N(\mathcal{Y})$, respectively. Then $\mathcal{X} = \{x_1, \ldots, x_{N(X)}\}$ and $\mathcal{Y} = \{y_1, \ldots, y_{N(Y)}\}$. We refer to source symbol x_n by its index $n \in \{1, \ldots, N(\mathcal{X})\}$ and side-information symbol y_m by its index $m \in \{1, \ldots, N(\mathcal{Y})\}$. Given the joint probability $p[n, m]$ of (x_n, y_m), $p[m] = \sum_{n=1}^{N(\mathcal{X})} p[n, m]$ is the probability of y_m. We again consider both optimal design subject to the convexity constraint and fast suboptimal design achieved by partitioning $\mathcal{X}$ into $\hat{N}(\mathcal{X})$ cells and $\mathcal{Y}$ into $\hat{N}(\mathcal{Y})$ cells. As in the previous discussion, $\hat{N}(\mathcal{X}) = N(\mathcal{X})$ and $\hat{N}(\mathcal{Y}) = N(\mathcal{Y})$ are required to guarantee optimal design subject to the convexity constraint.

Like the ECSQ encoder of Section III, the SECSQ encoder partitions $\mathcal{X}$ into convex codecells. Thus, finding the best code that meets the convexity constraint is equivalent to designing the best partition with convex elements. Further, since the expected distortion and expected rate for an encoder defined by partition $\mathcal{P}$ of $\mathcal{X}$ are additive over the codecells of $\mathcal{P}$, we can once again define a partial RD graph to make optimal partition design equivalent to a shortest path problem. The only difference between the partial RD graph for SECSQ and the partial RD graph for ECSQ lies in the weight calculations.

The expected distortion of the SECSQ defined by partition $\mathcal{P}$ is $D_S(\mathcal{P}) = \sum_{C\in\mathcal{P}} d_S(C)$. Here

$$d_S(C) = \sum_{n\in C} \sum_{m=1}^{N(\mathcal{Y})} p[n, m] d(x_n, \mu(C, m))$$

where $\mu(C, m)$ is the optimal codeword for codecell C and side information y_m. When $d(x, \hat{x}) = (x - \hat{x})^2$, the centroid $\mu(C, m) = \sum_{n\in C} p[n, m] x_n / p(C, m)$ of C with respect to pmf $p[n|m] = p[n, m]/p[m]$ yields the optimal performance; here $p(C, m) = \sum_{n\in C} p[n, m]$.

In calculating the partial rate, we assume the use of a rate-$H(X|Y)$ Slepian–Wolf code to describe X to a decoder that knows Y [32]. While rate $H(X\,|\,Y)$ may not be achievable with probability of error $P_e = 0$, we can design codes that approach rate $H(X|Y)$ arbitrarily closely and achieve $P_e < \epsilon$ for any $\epsilon > 0$ [33]. (Given assumptions of a finite alphabet and bounded distortion measure, the increase in distortion caused by a nonzero error probability can be made arbitrarily small.) Thus, the expected rate of the SECSQ defined by partition $\mathcal{P}$ is $R_S(\mathcal{P}) = \sum_{C\in\mathcal{P}} r_S(C)$ with partial rate

$$r_S(C) = -\sum_{m=1}^{N(\mathcal{Y})} p(C, m) \log p(C\,|\,m)$$

where $p(C\,|\,m) = \sum_{n\in C} p[n\,|\,m]$.

The SECSQ optimization criterion is the Lagrangian

$$J(\mathcal{P}, \lambda) = D_S(\mathcal{P}) + \lambda R_S(\mathcal{P}).$$

The weight on arc (u, v) of the partial RD graph is therefore defined as $d_S(C_{(u,v]}) + \lambda r_S(C_{(u,v]})$. Given these modified weights, the design procedure is identical to the ECSQ design procedure.

Other Side-Information Codes

The inclusion of decoder side-information in fixed-rate conventional scalar quantizers and fixed- and variable-rate MDSQs, MRSQs, and restricted MDSQs proceeds similarly. In each case, the design criterion and algorithm for the side-information code are identical to those of the code without side information. The only differences are the partial rate and distortion calculations used to calculate arc weights in the partial RD graph. Without side information, codecell C has a single codeword and contributes

$d(C) = \sum_{n \in C} p[n] d(x_n, \mu(C))$ to its partition's distortion; with side information, codecell C has $N(\mathcal{Y})$ codewords and contributes $d_S(C) = \sum_{n \in C} \sum_{m=1}^{N(\mathcal{Y})} p[n,m] d(x_n, \mu(C,m))$ to its partition's distortion. Without side information, variable-rate coding rates are entropies and conditional entropies; with side information, those entropies and conditional entropies are (further) conditioned on the decoder side information. The rates for fixed-rate codes do not change as a result of decoder side information.

VII. Computational Complexity

The proof of correctness in each section demonstrates that each algorithm yields a globally optimal solution subject to the constraint of codecell convexity. We next summarize the complexities of these algorithms.

All design algorithms consist of two steps: the preprocessing step, where we calculate the partial rates and distortions needed to define the edge weights, and the shortest path step, where we find the shortest path through the graph. The complexity of the first step depends only on the size of the source alphabet (or its reduction) and distortion measure, while the complexity of the second step also depends on the code type. The following results bound the complexity of the preprocessing and shortest path steps, respectively.

Theorem 1: For non-mean-square error (MSE) distortion measures, the preprocessing step takes $O(\hat{N}^2 N)$ time and $O(\hat{N}^2)$ space. For MSE distortion, the preprocessing step takes $O(\hat{N})$ space and time.

Proof: For a general distortion measure, the preprocessing step involves computing and storing the partial rates and distortion. There are $O(\hat{N}^2)$ values to compute, computing each requires $O(N)$ operations in general, which gives the first result.

In the MSE case, we do not compute or store the edge weights; given the source pmf restricted to the coarse grid, we merely calculate the source pmf's cumulative moments of order $1, 2$, and 3. From these, we can compute any partial rate or distortion on the fly in constant time. The moments are $3(\hat{N}+1)$ real numbers that can be computed in linear time and stored in an array. □

Theorem 2: The shortest path step complexities are:

VR-MDSQ: $O((\hat{N}+1)^M)$ space, $O(M(2(\hat{N}+1))^{M+1})$ time.

FR-MDSQ: $O((\hat{N}+1)^M \prod_{i=1}^M (d_i+1))$ space, $O(M(2(\hat{N}+1))^{M+1} \prod_{i=1}^M (d_i+1))$ time, where d_i is the fixed length for side partition i.

VR-MRSQ: $O(\hat{N}^2)$ space, $O(M\hat{N}^3)$ time.

FR-MRSQ: $O((\hat{N}+1)^{2M-1} \prod_{i=1}^M (d_i+1))$ space, $O(M2^M(\hat{N}+1)^{2M} \prod_{i=1}^M (d_i+1))$ time.

Restricted MDSQ (possibly with some FR descriptions):

$O((\hat{N}+1)^{M+p_d} \prod_{i=1}^M (d_i+1))$ space,
$O(M2^M(\hat{N}+1)^{M+1+p_d} \prod_{i=1}^M (d_i+1))$ time,

where d_i is the fixed length for side partition i or 0 for VR side partitions, and

p_d is the number of partitions with dependents.

These complexities apply whether the code designer has side information or not.

Proof: For all algorithms except MRSQ-VR, the shortest path step involves first generating and storing the topologically sorted list of vertices for the partial RD graph, computing the edge weights, and then finding the shortest path. If the graph has V vertices and E edges, then the first step takes $O(V)$ space and $O(\max(V,E))$ time. The second step takes constant space and time proportional to $O(E)$, but the proportionality factor is not a constant and depends on the algorithm. The third step takes $O(E)$ time. Thus, the space complexity is $O(V)$, while the time complexity needs to be settled in each case.

For variable-rate MDSQ, there are $(\hat{N}+1)^M$ vertices; each vertex has at most $\hat{N}$ outgoing arcs, thus there are $O((\hat{N}+1)^{M+1})$ edges. The number of edges dominates the number of vertices. In general, up to 2^{M-1} codecells are generated for each edge, and determining the boundaries of each codecell requires examining at most M coordinates.

For fixed-rate MDSQ, there are $(\hat{N}+1)^M \prod_{i=1}^M (d_i+1)$ vertices; each vertex has $O(\hat{N})$ edges. Again, there are more edges than vertices. As with MDSQ, computing the weight of each edge requires $O(M2^{M-1})$ steps.

In a restricted MDSQ, each partition may have one, two, or three labels. Precisely, each partition has one extra label per feature, where the features considered are: "fixed-rate" and "has dependents." The basic label and the label for partitions with dependents ranges from 0 to $\hat{N}$. The fixed-rate label ranges from 0 to d_i. Therefore, the number of vertices increases to $O((\hat{N}+1)^{M+p_d}) \prod_{i=1}^M (d_i+1)$. The number of outgoing edges per vertex is again $O(\hat{N})$; computing each edge weight still requires $O(M2^{M-1})$ steps.

The fixed-rate MRSQ result relies on the restricted MDSQ result. In this case, every partition is fixed rate, and all but the finest one have dependents.

In variable-rate MRSQ, we iterate over M resolution levels. At each level, we run an all-pairs shortest path algorithm in a 1DSQ-like graph with $\hat{N}$ vertices, which takes $O(\hat{N}^3)$ time, and store the resulting costs, which takes $O(\hat{N}^2)$ space. The costs become the weights for the next level up, and are overwritten with new costs after that level completes. □

VIII. Codecell Convexity: Conditions for Optimality

The preceding sections describe algorithms for choosing the optimal quantizer among all quantizers with convex codecells. We here explore the consequences of the convexity constraint under the assumption that $d(x,\hat{x}) = \rho(|x-\hat{x}|)$ for some nondecreasing $\rho : [0,\infty) \to [0,\infty)$. Briefly put, the conclusions are as follows. Under reasonably mild constraints on ρ, convex codecells are sufficient for optimality in conventional scalar quantization and ECSQ. In contrast, we give examples to demonstrate that even for the squared error distortion measure $(\rho(x) = x^2)$, convex codecells preclude optimality for MRSQ, MDSQ, and side-information scalar quantizers for some sources at some rates.

The benefit of the convexity constraint is the low-complexity algorithms that it enables. The cost is the performance degradation that results when convexity precludes optimality. A more precise understanding of this tradeoff is an important topic for future research. For example, demonstrating the existence of examples where convexity precludes optimality does not prove that convex codecells are *never* optimal for these code types. In fact, for MRSQ it seems to be far easier to design examples for which convex codecells are optimal than to design examples requiring nonconvex codecells. Further, even when convexity precludes optimality, the gap between the performance of the constrained optimum and the unconstrained optimum is not well understood.

The discussion that follows begins with fixed-rate and entropy-constrained conventional scalar quantization and then treats first fixed-rate and entropy-constrained MRSQ, MDSQ, and restricted MDSQ and then decoder side information.

Conventional Scalar Quantizers

Let $\mathcal{D}^{\text{fr}} = \{(\log|\mathcal{P}|, D(\mathcal{P}))\}$ and $\mathcal{D}^{\text{vr}} = \{(R(\mathcal{P}), D(\mathcal{P}))\}$ describe the sets of expected rate–distortion points achievable through fixed-rate and entropy-constrained scalar coding (without decoder side-information) on pmf $\{p[n]\}_{n=1}^{N}$. A conventional fixed-rate scalar quantizer (SQ) that achieves a point on the lower boundary $D_1^{\text{fr}}(R) = \inf_{\mathcal{P}:\log|\mathcal{P}|\le R} D(\mathcal{P})$ of $\mathcal{D}^{\text{fr}}$ is *optimal* in the sense that it has the lowest possible distortion among all fixed-rate conventional scalar quantizers satisfying some rate constraint. Similarly, an ECSQ that achieves a point on the lower boundary $D_1^{\text{vr}}(R) = \inf_{\mathcal{P}:R(\mathcal{P})\le R} D(\mathcal{P})$ of $\mathcal{D}^{\text{vr}}$ is *optimal* in the sense that it has the lowest possible distortion among all ECSQs satisfying some rate constraint. While $D_1^{\text{fr}}(R)$ and $D_1^{\text{vr}}(R)$ are not convex in general and any point on the lower convex hull of $D_1^{\text{fr}}(R)$ and $D_1^{\text{vr}}(R)$ can be achieved through time sharing, the above definitions for optimal fixed-rate SQs and ECSQs does not require performance on the convex hull of $D_1^{\text{fr}}(R)$ or $D_1^{\text{vr}}(R)$; time-sharing strategies, while low in complexity, are not strict *scalar* codes.

Theorem 3 proves the optimality of convex codecells for fixed-rate scalar quantizers when ρ is nondecreasing. The result is a very slight generalization of [34, Lemma 6.2.1] (which treats ρ increasing) and the earlier [35] (which treats $\rho(x) = x^2$). Our proof differs from the ones that precede it; we generalize this approach in the discussion that follows.

Theorem 3: Given a pmf $\{p[1], \ldots, p[N]\}$ and a distortion measure $d(x, \hat{x}) = \rho(|x - \hat{x}|)$ for some nondecreasing function $\rho : [0, \infty) \to [0, \infty)$, any point $(R, D_1^{\text{fr}}(R)) \in \mathcal{D}^{\text{fr}}$ is achievable by a fixed-rate SQ with convex codecells.

Proof: We prove that for any partition $\mathcal{P} = \{c_1, \ldots, c_K\}$ with optimal codewords $\mu(c_1) \le \cdots \le \mu(c_K)$, there exists another partition $\mathcal{P}^\star$ with convex codecells $\{c_k^\star\}_{k=1}^K$ that satisfies

$$\sum_{k=1}^{K} \sum_{n \in c_k^\star} p[n] d(x_n, \mu(c_k)) \le D(\mathcal{P}).$$

Since $(R, D_1^{\text{fr}}(R)) \in \mathcal{D}^{\text{fr}}$ implies, by definition of $\mathcal{D}^{\text{fr}}$, the existence of an encoder partition $\mathcal{P}$ such that $\log|\mathcal{P}| \le R$ and $D(\mathcal{P}) = D_1^{\text{fr}}(R)$, this observation gives the desired result.

For each $k, l \in \{1, \ldots, K\}$, let

$$c'_{k,l} = \begin{cases} \{x : d(x, \mu(c_k)) < d(x, \mu(c_l))\}, & \text{if } l < k \\ \{x : d(x, \mu(c_k)) \le d(x, \mu(c_l))\}, & \text{if } l \ge k. \end{cases}$$

Then $d(x, \hat{x}) = \rho(|x - \hat{x}|)$ and ρ nondecreasing imply that the $c'_{k,l}$ are half-lines. For example, if ρ is strictly increasing, then $c'_{k,l} = \{x : x < (\mu(c_k) + \mu(c_l))/2\}$ for each $l < k$ and $c'_{k,l} = \{x : x \ge (\mu(c_k) + \mu(c_l))/2\}$ for each $l \ge k$. The set $c_k^\star = \cap_{l=1}^K c'_{k,l}$ describes all x for which $\mu(c_k)$ is the closest codeword, with ties broken in favor of the smallest codeword. Since the $c'_{k,l}$ are half-lines, each $c_k^\star$ must be an interval. Finally, $\mathcal{P}^\star = \{c_k^\star\}_{k=1}^K$ partitions $\{1, \ldots, N\}$ (every x_n has a unique closest codeword) and minimizes expected distortion, giving the desired result. □

We make two modifications in generalizing Theorem 3 from fixed-rate to entropy-constrained codes. First, we consider only points on the convex hull of $\mathcal{D}^{\text{vr}}(R)$. Second, we assume that ρ is convex. The first constraint is practically motivated since the Lagrangian performance measure used for ECSQ design finds the lower convex hull of $\mathcal{D}^{\text{vr}}(R)$; it is also theoretically motivated since there exist points on $\mathcal{D}^{\text{vr}}(R)$ (but not on its convex hull) that cannot be achieved with convex codecells [36]. In the absence of the second constraint, codecell convexity may preclude encoder optimality for some source distributions. Codecell convexity for ECSQ on continuous source distributions under the squared error and rth-power distortion measures is noted without proof in [36, p. 418]. Interestingly, that observation also applies to optimal variable-rate codes with performance not on the lower convex hull of achievable rates.

Theorem 4: Given a pmf $\{p[1], \ldots, p[N]\}$ and a distortion measure $d(x, \hat{x}) = \rho(|x - \hat{x}|)$ where $\rho : [0, \infty) \to [0, \infty)$ is convex and nondecreasing, any point $(R, D_1^{\text{vr}}(R)) \in \mathcal{D}^{\text{vr}}$ on the lower convex hull of $D_1^{\text{vr}}(\cdot)$ is achievable by an ECSQ with convex codecells.

Proof: For any point $(R, D_1^{\text{vr}}(R)) \in \mathcal{D}^{\text{vr}}$, there exists a partition $\mathcal{P}$ with $R(\mathcal{P}) \le R$ and $D(\mathcal{P}) \le D_1^{\text{vr}}(R)$. We assume, without loss of generality, that $|\mathcal{P}| \le N < \infty$ since any partition with more than N codecells must include empty codecells and empty codecells cannot improve ECSQ performance. If $(R, D_1^{\text{vr}}(R)) \in \mathcal{D}^{\text{vr}}$ is on the lower convex hull of $D_1^{\text{vr}}(\cdot)$, then there exists a $\lambda > 0$ such that $\mathcal{P}$ minimizes $D(\mathcal{P}) + \lambda R(\mathcal{P})$ over all partitions on $\{1, \ldots, N\}$, as discussed in Section III. Let $\mathcal{P} = \{c_1, \ldots, c_K\}$ $(K \le N)$ with codecell probabilities $p(c_1), \ldots, p(c_k)$ and optimal codewords $\mu(c_1) \le \cdots \le \mu(c_K)$. We next construct a convex codecell partition $\mathcal{P}^\star = \{c_k^\star\}_{k=1}^K$ that satisfies

$$\sum_{k=1}^{K} \sum_{n \in c_k^\star} p[n](d(x_n, \mu(c_k)) + \lambda(-\log p(c_k))) \le D(\mathcal{P}) + \lambda R(\mathcal{P}).$$

While this property is sufficient to prove the desired result, we further note that

$$R(\mathcal{P}^\star) = \sum_{k=1}^{K} -p(c_k^\star) \log p(c_k^\star) \le \sum_{k=1}^{K} -p(c_k^\star) \log p(c_k)$$

giving

$$D(\mathcal{P}^\star) + \lambda R(\mathcal{P}^\star) \le D(\mathcal{P}) + \lambda R(\mathcal{P}).$$

For any $x \in [x_1, x_N]$ and $c \in \{c_1, \ldots, c_K\}$, let $j(x,c) = d(x, \mu(c)) + \lambda(-\log p(c))$. Then for each $k, l \in \{1, \ldots, K\}$, let

$$c'_{k,l} = \begin{cases} \{x : j(x,c_k) < j(x,c_l)\}, & \text{if } l < k \\ \{x : j(x,c_k) \le j(x,c_l)\}, & \text{if } l \ge k. \end{cases}$$

From [36, Lemma 1], if $\rho(x)$ is convex and nondecreasing in x, then $\rho(|x - \hat{x}_1|) - \rho(|x - \hat{x}_2|)$ is monotonic in x. Since $\rho(|x-\hat{x}_1|) - \rho(|x-\hat{x}_2|)$ is monotonic in x, $j(x,c_k) - j(x,c_l)$ is monotonic in x for each (k,l). As a result, each nonempty $c'_{k,l}$ is a half-line. The set $c^\star_k = \cap_{l=1}^K c'_{k,l}$ describes all x for which $\mu(c_k)$ is the "closest" codeword by this modified nearest neighbor distortion measure, with ties broken in favor of the smallest codeword. Since the $c'_{k,l}$ are half-lines, each $c^\star_k$ must be an interval. Again $\mathcal{P}^\star = \{c^\star_k\}_{k=1}^K$ partitions $\{1, \ldots, N\}$, and thus we have the desired result. □

If ρ is nondecreasing and not convex, then codecell convexity may preclude encoder optimality for some source distributions since guaranteeing a unique solution to $\rho(|x - \hat{x}_1|) + \lambda r_1 = \rho(|x-\hat{x}_2|) + \lambda r_2$ requires monotonicity of $\rho(|x-\hat{x}_1|) - \rho(|x-\hat{x}_2|)$ in x, which in turn requires convexity of ρ, as shown next. For any fixed $a < b$, let

$$\begin{aligned} f(x) &= \rho(|x-a|) - \rho(|x-b|) \\ &= \begin{cases} \rho(a-x) - \rho(b-x), & \text{if } x < a \\ \rho(x-a) - \rho(b-x), & \text{if } a \le x \le b \\ \rho(x-a) - \rho(x-b), & \text{if } x > b. \end{cases} \end{aligned}$$

If ρ is nondecreasing, then $f(x)$ is nondecreasing for all $x \in [a,b]$. This leaves two possibilities: either $f(x)$ is constant on $[a,b]$ or $f(a) < f(b)$. The first case cannot be true for all $[a,b]$ (that is, all codewords) unless $\rho(x) = c$ for all x, which is a convex function. In the second case, monotonicity of $f(x)$ requires that $\rho(a-x) - \rho(b-x)$ is nondecreasing for all $x < a$ and that $\rho(x-a) - \rho(x-b)$ is nondecreasing for all $x > b$. The second condition gives $\rho(x-a) - \rho(x-b) \le \rho(x+\Delta-a) - \rho(x+\Delta-b)$ or

$$\rho(x+\Delta-b) - \rho(x-b) \le \rho(x+\Delta-a) - \rho(x-a)$$

for all $x > b$ and any $\Delta \ge 0$. While achieving this result for some (a,b) does not require the convexity of ρ, achieving this result for *all* a, b requires the convexity of ρ. Thus, when ρ is not convex, there exists a pmf with optimal codewords a and b such that $f(x)$ is not monotonic. The existence of such a pmf opens the door for an optimal code that requires nonconvex codecells.

MRSQ, MDSQ, and Restricted MDSQ

In MRSQ, MDSQ, and restricted MDSQ, codecell convexity means that all codecells of all partitions are convex. To achieve this constraint in MDSQ, it suffices to choose side partitions $\mathcal{P}_{0^i-1 1 0^{M-i}}$ with convex codecells.

We begin by considering the simplest form of (restricted) MDSQ, namely MRSQ. The rate–distortion points achievable by fixed-rate-$\underline{R}$ and entropy-constrained MRSQs are

$$\begin{aligned} \mathcal{D}^{\text{fr}}(\underline{R}) &= \{(R_i, D(\mathcal{P}_i))|_{i=1}^M : \underline{\mathcal{P}} \in \boldsymbol{P}^{\text{fr}}(\underline{R})\} \\ \mathcal{D}^{\text{vr}} &= \{(R(\mathcal{P}_i \,|\, \mathcal{P}_{i-1}), D(\mathcal{P}_i))|_{i=1}^M : \underline{\mathcal{P}} \in \boldsymbol{P}^{\text{vr}}\}\} \end{aligned}$$

respectively, where

$$\begin{aligned} \boldsymbol{P}^{\text{fr}}(\underline{R}) = \{\underline{\mathcal{P}} : \mathcal{P}_M \succeq \cdots \succeq \mathcal{P}_1 \\ \wedge \max_{C \in \mathcal{P}_{i-1}} \log |\mathcal{P}_i(C)| \le R_i \; \forall i\} \\ \boldsymbol{P}^{\text{vr}} = \{\underline{\mathcal{P}} : \mathcal{P}_M \succeq \cdots \succeq \mathcal{P}_1\}. \end{aligned}$$

An MRSQ that achieves a point on a lower boundary of $\mathcal{D}^{\text{fr}}(\underline{R})$ for any nonnegative $\underline{R}$ (here R_i is the incremental rate of resolution i) or $\mathcal{D}^{\text{vr}}$ is in some sense optimal. (In fixed-rate-$\underline{R}$ coding, each lower boundary describes the minimal value of some D_j subject to constraints on $\{D_i\}_{i \ne j}$. In entropy-constrained coding, each lower boundary describes the minimal value of some D_i or R_i subject to constraints on the remaining rates and distortions.) While the lower boundaries are not convex in general, our design technique focuses on achieving points on the lower convex hull of $\mathcal{D}^{\text{fr}}(\underline{R})$ and $\mathcal{D}^{\text{vr}}$. For any M successively refined partitions $\{\mathcal{P}^\star_i\}_{i=1}^M$ that together achieve a point on the lower convex hull of $\mathcal{D}^{\text{fr}}(\underline{R})$, there exists a positive vector $\underline{\nu} = \{\nu_i\}_{i=1}^M$ for which $\{\mathcal{P}^\star_i\}_{i=1}^M$ minimizes the Lagrangian

$$J^{\text{fr}}(\underline{\mathcal{P}}, \underline{R}, \underline{\nu}) = \sum_{i=1}^M \nu_i D(\mathcal{P}_i)$$

over all $\underline{\mathcal{P}} \in \boldsymbol{P}^{\text{fr}}(\underline{R})$. Similarly, any M successively refined partitions that together achieve a point on the lower convex hull of $\mathcal{D}^{\text{vr}}$ minimize the Lagrangian

$$J^{\text{vr}}(\underline{\mathcal{P}}, \underline{\nu}, \underline{\lambda}) = \sum_{i=1}^M [\nu_i D(\mathcal{P}_i) + \lambda_i R(\mathcal{P}_i | \mathcal{P}_{i-1})]$$

for some positive vectors $\underline{\nu} = \{\nu_i\}_{i=1}^M$ and $\underline{\lambda} = \{\lambda_i\}_{i=1}^M$.

The algorithm described in Section V finds the optimal MRSQ among MRSQs with convex codecells *at all resolutions*. That is, the design algorithm considers only MRSQs with convex partitions $\mathcal{P}_1, \ldots, \mathcal{P}_M$. Unfortunately, there exist pmfs for which the constraint of codecell convexity in $\mathcal{P}_1$ precludes optimality even for points on the lower convex hull of $\mathcal{D}^{\text{fr}}(\underline{R})$ with $d(x,\hat{x}) = \rho(|x-\hat{x}|)$ and ρ convex. One such example follows. Due to the popularity of the squared-error distortion measure for practical coding applications, we here use $d(x,\hat{x}) = (x-\hat{x})^2$. We can construct similar examples for other distortion measures.

Example 5: Let $d(x,\hat{x}) = (x-\hat{x})^2$, and consider pmf $\{1/8, 1/8, 3/8, 3/8\}$ on alphabet $\{20, 40, 60, 140\}$. Then for $M = 2$ there exists a point on the lower convex hull of $\mathcal{D}^{\text{fr}}((1,1))$ that cannot be achieved with codecell convexity. In particular, in order to minimize D_1 with $(R_1, R_2) = (1,1)$ and $D_2 = 0$, we must use nonconvex codecells at resolution 1. The optimal partitions (in terms of the symbol alphabet rather than the symbol indices) are $\mathcal{P}_1 = \{\{20,60\}, \{40,140\}\}$ and $\mathcal{P}_2 = \{\{20\}, \{60\}, \{40\}, \{140\}\}$. Fig. 15 shows the corresponding optimal 2RSQ codebook. The intuition here is as follows. By setting the incremental rates as $R_1 = R_2 = 1$, we fix the maximal depth of our binary tree-structured codebook to two. By further setting the second-resolution distortion D_2 to 0, we require each of our four alphabet symbols to occupy a different leaf of the binary codebook. The question that

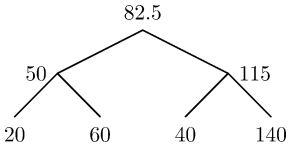

Fig. 15. An optimal fixed-rate 2RSQ codebook for which $\mathcal{P}_1$ requires nonconvex codecells. Given pmf $\{1/8, 1/8, 3/8, 3/8\}$ on alphabet $\{20, 40, 60, 140\}$, the optimal 2RSQ for all ν_1, ν_2 such that $\nu_1/\nu_2 < 0.02695$ must have nonconvex codecells at resolution 1. The codebook's resolution-1 partition is $\mathcal{P}_1 = \{\{20, 60\}, \{40, 140\}\}$ with $\mu(\{20, 60\}) = 50$ and $\mu(\{40, 140\}) = 115$.

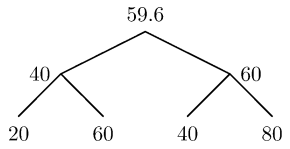

Fig. 16. An optimal entropy-constrained 2RSQ codebook with nonconvex codecells in $\mathcal{P}_1$. Given pmf $\{1/8, 3/8, 1/8, 3/8\}$ on alphabet $\{20, 40, 60, 80\}$, the optimal 2RSQ for all $(\lambda_1, \lambda_2, \nu_1, \nu_2)$ such that $\lambda_2 + 31.36\nu_1 < \lambda_1 < \lambda_2 + 5951.16\nu_1$ and ν_2 is sufficiently large to force D_2 to 0 must have nonconvex codecells at resolution 1. Here $\mathcal{P}_1 = \{\{20, 60\}, \{40, 80\}\}$ with $\mu(\{20, 60\}) = 40, \mu(\{40, 80\}) = 60$, $R(\mathcal{P}_1) = .1414$, and $R(\mathcal{P}_2|\mathcal{P}_1) = 1$.

remains, then, is which pairing of our four symbols into two groups of two minimizes the first-resolution distortion D_1. While it seems natural to group the symbols lexicographically, with symbols 20 and 40 descending from one first-resolution node and 60 and 140 from the other, this turns out not to be the optimal choice. The key here is that most of the probability lies in symbols 60 and 140 and these symbols are quite far apart. Pairing 20 with 60 and pairing 40 with 140 (see Fig. 15) gives lower expected distortion since it allows more accurate reproduction of the high probability events. □

The same problem can arise for entropy-constrained MRSQs achieving performance on the lower convex hull of $\mathcal{D}^{\text{vr}}$, as shown in Example 6.

Example 6: Consider $M = 2, d(x, \hat{x}) = (x - \hat{x})^2$, pmf $\{1/8, 3/8, 1/8, 3/8\}$, and alphabet $\{20, 40, 60, 80\}$. If $\lambda_2 + 31.36\nu_1 < \lambda_1 < \lambda_2 + 5951.16\nu_1$ and ν_2 is sufficiently large to force D_2 to 0, then achieving the optimal performance requires nonconvex codecells at resolution 1. The optimal partitions (again in terms of the symbol alphabet rather than the symbol indices) are $\mathcal{P}_1 = \{\{20, 60\}, \{40, 80\}\}$ and $\mathcal{P}_2 = \{\{20\}, \{60\}, \{40\}, \{80\}\}$. Fig. 16 shows the optimal 2RSQ codebook. The intuition in this case differs from that in the prior example. In this case, the argument is a bit more complicated. Setting $M = 2$ implies a depth-2 codebook, and making ν_2 sufficiently large to force D_2 to 0 implies that each source symbol must occupy a distinct leaf in that codebook. In this case, however, the tree structure need not be binary. Luckily, the space of possible solutions is sufficiently small, that we can test each possible solution. The key to our nonconvex solution is that, under the given constraints on λ and ν, the rate advantage associated with pairing the high-probability symbols (40 and 80) together and pairing the low-probability symbols (20 and 60) together outweighs the distortion cost associated with their greater distortion. □

While the preceding examples demonstrate that convexity sometimes precludes optimality, it is important to note that convex codecells are also optimal for some distributions. For example, violating the condition $\nu_1/\nu_2 < 0.02695$ in Example 5 or the condition $\lambda_2 + 31.36\nu_1 < \lambda_1 < \lambda_2 + 5951.16\nu_1$ in Example 6 yields examples where there exists an optimal code with convex codecells.

Since MRSQ is a special case of a restricted MDSQ, the possible suboptimality of the optimal code with convex codecells generalizes to MDSQ and restricted MDSQ as well. The result is the following theorem.

Theorem 5: Requiring codecell convexity in partition $\mathcal{P}_1$ of a fixed-rate or entropy-constrained MRSQ, MDSQ, or restricted MDSQ with $M > 1$ precludes optimality for some finite-alphabet sources, even when $d(x, \hat{x}) = \rho(|x - \hat{x}|)$ with $\rho : [0, \infty) \to [0, \infty)$ nondecreasing and convex.

The cost of convex codecells in MDSQ is exacerbated by the additional constraints it imposes on the size of the intersection partitions. For example, an MDSQ with d_i convex codecells in partition $\mathcal{P}_{(0^{i-1}10^{M-i})}, i \in \{1, \ldots, M\}$, has at most $(\sum_{i=1}^{M} d_i) + M - 1$ codecells in $\mathcal{P}_{(11\ldots1)}$. In contrast, when the codecells of each $\mathcal{P}_{(0^{i-1}10^{M-i})}$ are not constrained, partition $\mathcal{P}_{(11\ldots1)}$ can have as many as $\prod_{i=1}^{M} d_i$ codecells. This observation leads to two crude bounds on the cost of convexity in the central distortion $D_{(11\ldots1)}$ of a fixed-rate MDSQ. Let $D_1(R)$ be the first-order fixed-rate operational rate-distortion bound at rate R. Then $\nu_{(11\ldots1)} = 1$ and $\nu_\sigma = 0$ for all other σ implies that the difference between the performance of a fixed-rate MDSQ with convex codecells and the performance of a fixed-rate MDSQ with codecells that need not be convex is

$$D_1\left(\log\left(\left(\sum_{i=1}^{M} d_i\right) + M - 1\right)\right) - D_1\left(\sum_{i=1}^{M} \log d_i\right).$$

This difference in distortion increases with increasing rate and may be arbitrarily large. In contrast, the cost of convexity equals zero when $\nu_{(0^{i-1}10^{M-i})} = 1$ for all $i \in \{1, \ldots, M\}$ and $\nu_\sigma = 0$ for all other σ. While these bounds are not entirely meaningful (the 1DSQ design algorithm yields globally optimal codes in each of these cases) they do help build intuition about the algorithm's performance when the packet loss probability is either extremely low or extremely high.

The preceding discussion demonstrates the problem with assuming codecell convexity at the *coarsest* level partitions ($\mathcal{P}_1$ in MRSQ, $\mathcal{P}_{(0^{i-1}10^{M-i})}$ in MDSQ). The approach also suggests similar difficulties with codecell convexity at other levels since we can construct from our 2RSQ examples 3RSQ examples that mimic the observed behavior in resolutions 1 and 2, and so on. The approach lends little insight, however, into the finest level partition ($\mathcal{P}_M$ in an MRSQ, $\mathcal{P}_{(11\ldots1)}$ in MDSQ). We next show there exist distortion measures for which the assumption of codecell convexity at the finest level partition does not preclude optimality. Precisely, Theorem 6 proves this result for fixed-rate MRSQ with the squared-error distortion measure. The same argument extends immediately to entropy-constrained MRSQ and fixed-rate and entropy-constrained MDSQ

and restricted MDSQ. These results generalize the earlier observation of Vaishampayan, which treated the central partition of a fixed-rate 2DSQ under the squared error distortion measure [27, Sec.III-A].

Theorem 6: Given pmf $\{p[1],\dots,p[N]\}$ and distortion measure $d(x,\hat{x})=(x-\hat{x})^2$, any point $(\underline{R},\underline{D})\in\mathcal{D}^{\text{fr}}(\underline{R})$ that sits on the lower convex hull of $\mathcal{D}^{\text{fr}}(\underline{R})$ is achievable by a fixed-rate MRSQ with convex codecells in $\mathcal{P}_M$.

Proof: For any point $(\underline{R},\underline{D})\in\mathcal{D}^{\text{fr}}(\underline{R})$ on the lower convex hull of $\mathcal{D}^{\text{fr}}(\underline{R})$, $(\underline{R},\underline{D})\in\mathcal{D}^{\text{fr}}(\underline{R})$ implies the existence of partitions $\underline{\mathcal{P}}\in\boldsymbol{P}^{\text{fr}}(\underline{R})$ with $D(\mathcal{P}_i)=D_i$ for each $i\in\{1,\dots,M\}$. Further, since $(\underline{R},\underline{D})$ is on the lower convex hull of $\mathcal{D}^{\text{fr}}(\underline{R})$, there exists a nonnegative vector ν^M for which $\underline{\mathcal{P}}$ minimizes $\sum_{i=1}^M\nu_iD(\mathcal{P}_i)$ over all $\underline{\mathcal{P}}\in\boldsymbol{P}^{\text{fr}}(\underline{R})$. Label the codecells of each partition in $\underline{\mathcal{P}}$ as $\mathcal{P}_i=\{c_{i,1},\dots,c_{i,K(i)}\}$ $(K(i)\le 2^{R_i})$ with optimal codewords $\mu(c_{i,1}),\dots,\mu(c_{i,K(i)})$ for each $i\in\{1,\dots,M\}$. Without loss of generality, index the codecells of $\mathcal{P}_M$ so that

$$\mu(c_{M,1})\le\mu(c_{M,2})\le\cdots\le\mu(c_{M,K(M)}).$$

Since $\mathcal{P}_{i+1}$ refines $\mathcal{P}_i$ for each $i\in\{1,\dots,M-1\}$, let $a(i,k)$ denote the index of the resolution-i codecell from which codecell $c_{M,k}$ descends. (Here $i\in\{1,\dots,M\}$ and $a(M,k)=k$ for all $k\in\{1,\dots,K(M)\}$.) We next construct partitions $\underline{\mathcal{P}}^\star\in\boldsymbol{P}^{\text{fr}}(\underline{R})$ so that $\mathcal{P}_M^\star$ has convex codecells and the expected Lagrangian performance based on $\underline{\mathcal{P}}^\star$ is no worse than that based on $\underline{\mathcal{P}}$. For any $x\in[x_1,x_N]$ and any $c_{M,k}$ with $k\in\{1,\dots,K(M)\}$, let $j(x,c_{M,k})=\sum_{i=1}^M\nu_id(x,\mu(c_{i,a(i,k)}))$. Then for each $k,l\in\{1,\dots,K\}$, let

$$c'_{k,l}=\begin{cases}\{x:j(x,c_{M,k})<j(x,c_{M,l})\}, & \text{if } l<k\\ \{x:j(x,c_{M,k})\le j(x,c_{M,l})\}, & \text{if } l\ge k.\end{cases}$$

Since $d(x,\hat{x})=(x-\hat{x})^2$, the difference $j(x,c_{M,k})-j(x,c_{M,l})$ is linear in x for all $k\ne l$ (the quadratic terms cancel). As a result, each nonempty $c'_{k,l}$ is a half line, and the set $c^\star_{M,k}=\cap_{l=1}^K c'_{k,l}$, describes all x for which $\{\mu(c^\star_{i,a(i,k)})\}_{i=1}^M$ is the M-resolution reproduction with the best Lagrangian performance. (Ties are broken in favor of the smallest resolution-M codeword.) Since the $c'_{k,l}$ are half-lines, each $c^\star_{M,k}$ must be an interval. The partition $\mathcal{P}^\star_M=\{c^\star_{M,k}\}_{k=1}^M$ together with the ancestor relationships described by $a(\cdot,\cdot)$ describe a partition $\underline{\mathcal{P}}^\star\in\boldsymbol{P}^{\text{fr}}(\underline{R})$ with $\mathcal{P}^\star_i=\{c^\star_{i,k}\}_{k=1}^{K(i)}$ and $c^\star_{i,k}=\cup_{k'\in\{1,\dots,K(M)\}:a(i,k')=k}c^\star_{M,k'}$. (Here $|\mathcal{P}^\star_i|=|\mathcal{P}_i|$ for all i by construction.) Since $\underline{\mathcal{P}}$ minimizes the Lagrangian performance with respect to the given codebook

$$\sum_{k=1}^{K(M)}\sum_{n\in c^\star_{M,k}}p[n]\sum_{i=1}^M\nu_id(x_n,\mu(c_{i,a(i,k)}))\le\sum_{i=1}^M\nu_iD(\mathcal{P}_i)$$

giving the desired result. □

Generalizing Theorem 6 to MDSQ tells us that, for the squared-error distortion measure, the codecells of the partition $\mathcal{P}_{(11\dots1)}$ must be convex. This observation may turn out to be the key to unconstrained, globally optimal code design, at least in some cases. For example, in fixed-rate MDSQ design, combining the fixed-rate constraints with the convex-codecell constraint on partition $\mathcal{P}_{(11\dots1)}$ gives a bound on the maximal number of codecells in $\mathcal{P}_{(11\dots1)}$, which in turn constrains the number of intervals that may be observed in the nonconvex codecells of the side partitions from which $\mathcal{P}_{(11\dots1)}$ derives. Generalizing our algorithm to efficiently design MDSQs for which each codecell is a union of at most K convex subcells and the resulting intersection partition $\mathcal{P}_{(11\dots1)}$ meets the above constraint on the maximal number of codecells is a topic of ongoing research.

Side-Information Codes

Unfortunately, the codecell convexity constraint can also degrade the performance of codes with decoder side information. This problem can occur even in simple fixed-rate SQ with $d(x,\hat{x})=\rho(|x-\hat{x}|)$ for a well-behaved function ρ. The following example illustrates the problem.

Example 7: Consider $M=2, d(x,\hat{x})=(x-\hat{x})^2$, pmf $\{1/4,1/4,1/4,1/4\}$, and alphabet $\{20,40,60,80\}$. Suppose that the side information takes on values from $\{1,2\}$ with $\Pr(1|X=20)=\Pr(1\,|\,X=40)=\Pr(2\,|\,X=60)=\Pr(2\,|\,X=80)=0.999$. The side information is available only to the decoder. Then the optimal fixed-rate-1 SQ with decoder side information requires nonconvex codecells. In particular, $\mathcal{P}=\{\{20,60\},\{40,80\}\}$. □

Theorem 7: Requiring codecell convexity in fixed-rate or variable-rate SQ, MRSQ, MDSQ, or restricted MDSQ with decoder side information precludes optimality for some finite-alphabet sources, even when $d(x,\hat{x})=\rho(|x-\hat{x}|)$ with $\rho:[0,\infty)\to[0,\infty)$ nondecreasing and convex.

IX. Experimental Results

A. Gaussian Data

Fig. 17 compares the performance of the proposed technique (pluses) with the experimental results printed in [27] (squares) on a Gaussian data set. The algorithm in [27] combines iterative descent with an index assignment algorithm. In each case, we design a fixed-rate 2DSQ with 2 bits per description and plot the code's central distortion as a function of its average side distortion. The results are comparable for the small range of side distortions where they overlap. The region of overlap is very small, as our algorithm finds many points with low average side distortion (but high central distortion), while index assignment finds some points with low central distortion, but high average side distortion.

The explanation for this lies in the convexity constraint. Given four convex codecells per side partition, the central partition has at most $4+4-2+1=7$ codecells. Running our algorithm for a seven-cell fixed-rate 1DSQ confirms that the lowest achievable distortion for this scenario is about 0.044 (which is the lowest central distortion achieved by the 2DSQ optimization). The fact that [27] achieves central distortions as low 0.014 proves that the index assignment method can generate nonconvex side partitions. For completeness, the largest number of central codecells achievable with nonconvex side partitions is $4\cdot4=16$, and those could be used to design a code with distortion as low as 0.0095.

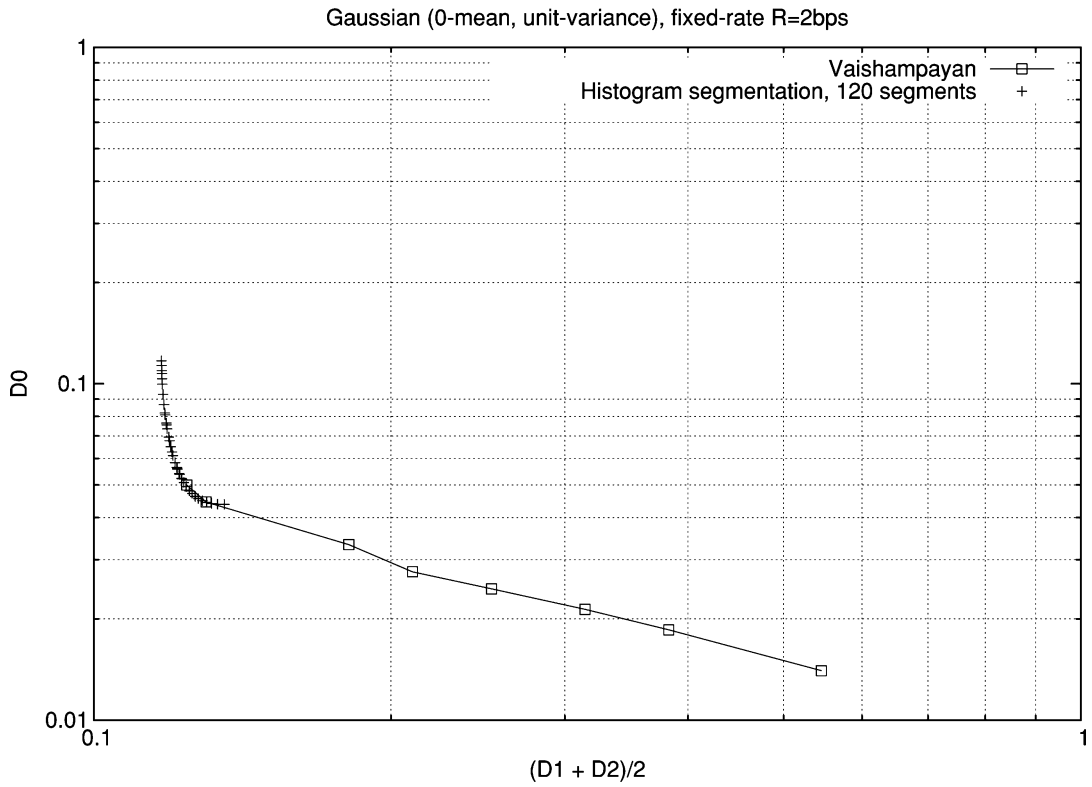

Fig. 17. Histogram segmentation versus index assignment, for a fixed-rate 2DSQ with 2 bits per description of Gaussian data.

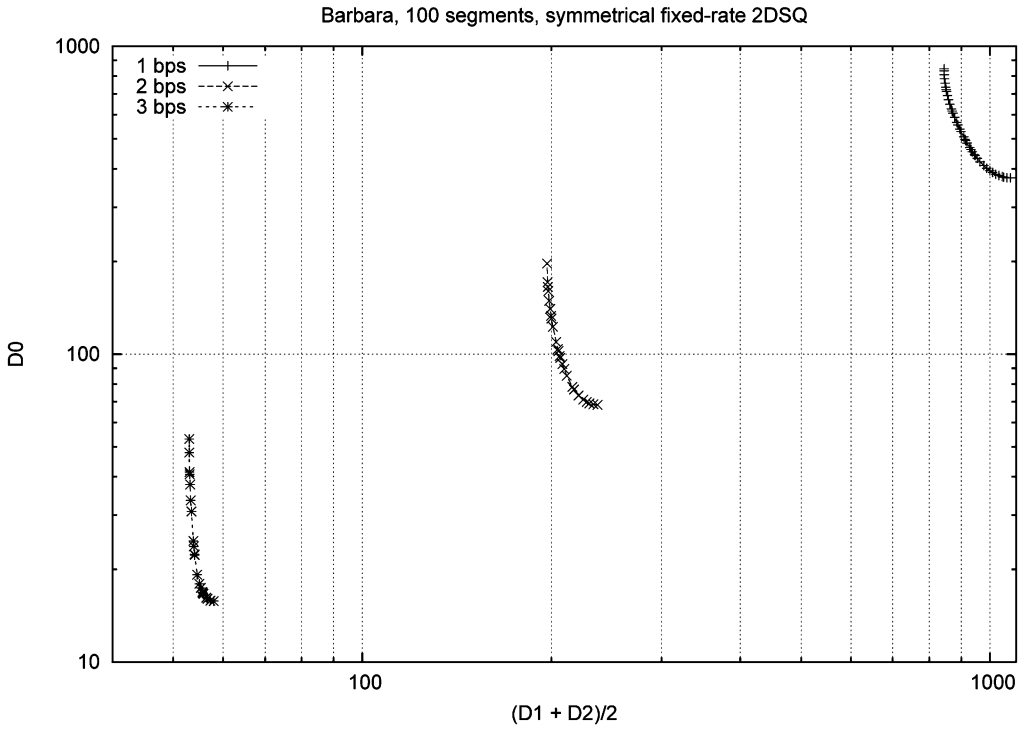

Fig. 18. Experimental results for a fixed-rate 2DSQ on the image Barbara.

B. Image Data

The results that follow show experimental results on the well-known Barbara image.

Fig. 18 shows the central distortion as a function of the average side distortion for 2DSQs with equal rates of 1, 2, and 3 bits per symbols (bps) in each description. Increasing the rate per description decreases all expected distortions. Decreasing the central distortion causes the average side distortion to increase.

Fig. 19 shows the experimental results for a fixed-rate 3DSQ with 1 bit per description. The graph shows the largest and

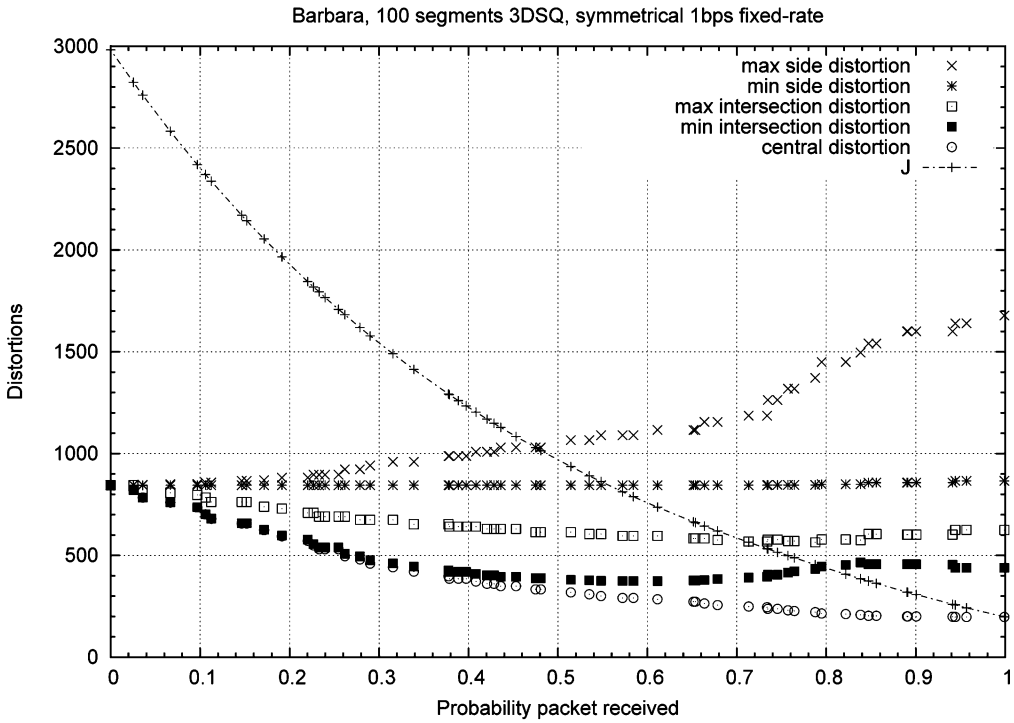

Fig. 19. Experimental results for a fixed-rate 3DSQ with 1 bps per description on the image Barbara.

smallest distortion values achieved by side partitions (labeled maximal and minimal side distortions), the largest and smallest distortion values achieved by intersection partitions (labeled maximal and minimal intersection distortions), the distortion of the central partition, and the Lagrangian performance J as a function of the probability that each packet is received. When $p = 0$, no description ever arrives at the decoder. As a result, the decoder's expected distortion, J, equals the source variance 2982. In this case, the partitions are optimized independently. As a result, each achieves an expected distortion of 845, which is the optimal distortion for a rate-1 1DSQ on the given source. Since the partitions are identical, there is no advantage to receiving more than one description, and the distortions associated with receiving two or three descriptions (the "intersection" and "central" distortions, respectively) are identical to the distortion for one description (the "side" distortion) As p increases from zero, so too does the probability of receiving more than one description. As a result, the partitions begin to differentiate in order to improve the distortion associated with receiving more than one description. We observe a slow rise in the distortion of any one description accompanied by a decrease in the distortions for any two or all three descriptions. By $p = 1/3$, the probabilities of receiving zero, one, two, and three descriptions are $8/27, 4/9, 2/9$, and $1/27$, respectively, giving an expected number of received descriptions of one. The average distortion from one description is 910, while the average distortion for two and three descriptions decreases to 569 and 419, respectively. (The distortion for zero descriptions always equals the variance, 2982.)

As p continues to increase, so too does the expected number of received descriptions. By $p = 2/3$, zero, one, two, and three descriptions are received with probabilities $1/27, 2/9, 4/9, 8/27$, respectively, and the expected number of received descriptions is two. The average distortions achieved with one, two, and three descriptions are $1034, 496$, and 264.

Finally, when $p = 1$, the probability of receiving fewer than three descriptions equals 0, and the expected distortion observed at the decoder equals 197. While the decoder has received a total of 3 bps, the achieved distortion equals the optimal distortion of a rate-2 1DSQ. This is, again, the price of convex codecells: the intersection of three rate-1 partitions with convex codecells can form at most four distinct cells. The given code achieves the optimal performance subject to this constraint on the central partition size.

The points $p = 0$ and $p = 1$ are the only places where we can tightly evaluate the cost of codecell convexity. We cannot find this quantity for any other values of p since the problem of optimal MDSQ design (without the convexity constraint) remains unsolved.

These results use the fixed-rated MDSQ design algorithm on an approximate coarse grid of 100 segments (the original image had 239 gray levels). This is only an approximation, but in our judgment a good one: we have experimented with 25, 50, 75, and 100 segments and found out that, while more segments lead to more data points on the convex hull, the performance does not change much. That is, the 25-segment results closely interpolate between the more numerous 50-segment results, and so on.

ACKNOWLEDGMENT

The authors are enormously grateful to anonymous reviewer A, whose detailed and insightful suggestions improved this document significantly.

REFERENCES

[1] J. D. Bruce, "Optimum Quantization," Ph.D. dissertation, MIT, Cambridge, MA, May 1964.

[2] D. K. Sharma, "Design of absolutely optimal quantizers for a wide class of distortion measures," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 6, pp. 693–702, Nov. 1978.
[3] X. Wu, "Algorithmic Approach to Mean-Square Quantization," Ph.D. dissertation, University of Calgary, Calgary, AB, Canada, 1988.
[4] X. Wu and K. Zhang, "Quantizer monotonicities and globally optimal scalar quantizer design," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 1049–1053, May 1993.
[5] D. Muresan and M. Effros, "Quantization as histogram segmentation: Globally optimal scalar quantizer design in network systems," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 2002, pp. 302–311.
[6] M. Effros and D. Muresan, "Codecell contiguity in optimal scalar quantizers," in *Proce. Data Compression Conf.*, Snowbird, UT, Mar. 2002, pp. 312–321.
[7] D. Muresan and M. Effros, "Data Compression Method and System Using Globally Optimal Scalar Quantization," U.S. Patent 6771831, Provisional application filed November 2001. Non-Provisional application filed November 2002. Patent Granted August 2004.
[8] S. Dumitrescu and X. Wu, "On Optimal Multi-Resolution Scalar Quantization," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 2002, pp. 322–331.
[9] S. Dumitrescu and X. Wu, "Optimal multiresolution quantization for scalable multimedia coding," in *Proc. Information Theory Workshop*, Bangalore, India, Oct. 2002, pp. 139–142.
[10] S. Dumitrescu and X. Wu, "Algorithms for optimal multi-resolution quantization," *J. Algorithms*, vol. 50, pp. 1–22, 2004.
[11] S. Dumitrescu and X. Wu, "Lagrangian global optimization of two-description scalar quantizers," in *Proc. IEEE Int. Symp. Information Theory*, Chicago, IL, Jun./Jul. 2004, p. 300.
[12] S. Dumitrescu, X. Wu, and G. Bahl, "Fast algorithms for optimal two-description scalar quantizer design," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 2004, pp. 42–51.
[13] S. Dumitrescu and X. Wu, "Optimal two-description scalar quantizer design," *Algorithmica*, vol. 41, no. 4, pp. 269–287, Feb. 2005.
[14] N. Moayeri, D. L. Neuhoff, and W. Stark, "Fine-coarse vector quantization," *IEEE Trans. Signal Process.*, vol. 39, no. 7, pp. 1503–1515, July 1991.
[15] P. A. Chou and T. Lookabaugh, "Locally optimal variable-to-variable length source coding with respect to a fidelity criterion," in *Proc. IEEE Int. Symp. Information Theory*, Budapest, Hungary, Jun. 1991, p. 238.
[16] M. Effros, P. A. Chou, and R. M. Gray, "Variable dimension weighted universal vector quantization and noiseless coding," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 1994, pp. 2–11, IEEE.
[17] Z. Xiong, C. Herley, K. Ramchandran, and M. T. Orchard, "Flexible time segmentations for time-varying wavelet packets," *IEEE Trans. Signal Process.*, vol. 3, no. 1, pp. 9–12, Jan. 1994.
[18] P. Prandoni, M. Goodwin, and M. Vetterli, "Optimal time segmentation for signal modeling and compression," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Munich, Germany, Apr. 1997, vol. 3, pp. 2029–2032.
[19] G. M. Schuster and A. K. Katsaggelos, "An optimal segmentation encoding scheme in the rate distortion sense," in *Proc. IEEE Int. Symp. Circuits and Systems*, Atlanta, GA, May 1996, vol. 2, pp. 640–643.
[20] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press/ McGraw-Hill, 1990.
[21] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy-constrained vector quantization," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 37, no. 1, pp. 31–42, Jan. 1989.
[22] S. Herman and K. Zeger, "Variable fanout trimmed tree-structured vector quantization for multirate channels," in *Proc. IEEE Int. Symp. Information Theory and Its Applications*, Victoria, BC, Canada, Sep. 1996, vol. 1, pp. 417–421.
[23] H. Brunk and N. Farvardin, "Fixed-rate successively refinable scalar quantizers," in *Proc. Data Compression Conf.*, Snowbird, UT, Apr. 1996, pp. 250–259.
[24] H. Jafarkhani, H. Brunk, and N. Farvardin, "Entropy-constrained successively refinable scalar quantization," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 1997, pp. 337–346.
[25] M. Effros, "Practical multi-resolution source coding: TSVQ revisited," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 1998, pp. 53–62.
[26] M. Effros and D. Dugatkin, "Multi-resolution vector quantization," *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 3130–3145, Dec. 2004.
[27] V. A. Vaishampayan, "Design of multiple description scalar quantizers," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 821–834, May 1993.
[28] V. A. Vaishampayan and J. Domaszewicz, "Design of entropy-constrained multiple description scalar quantizers," *IEEE Trans. Inf. Theory*, vol. 40, no. 1, pp. 245–250, Jan. 1994.
[29] M. Fleming and M. Effros, "Generalized multiple description vector quantization," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 1999, pp. 3–12.
[30] M. Fleming, Q. Zhao, and M. Effros, "Network vector quantization," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1584–1604, Aug. 2004.
[31] R. M. Gray and A. D. Wyner, "Source coding for a simple network," *Bell Syst. Tech. J.*, vol. 53, no. 9, pp. 1681–1721, Nov. 1974.
[32] D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 4, pp. 471–480, Jul. 1973.
[33] Q. Zhao and M. Effros, "Lossless and near-lossless source coding for multiple access networks," *IEEE Trans. Inf. Theory*, vol. 49, no. 1, pp. 112–128, Jan. 2003.
[34] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer, 1992.
[35] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 129–136, Mar. 1982, Previously an unpublished Bell Laboratories Technical Note (1957).
[36] A. György and T. Linder, "On the structure of optimal entropy-constrained scalar quantizers," *IEEE Trans. Inf. Theory*, vol. 48, no. 2, pp. 416–427, Feb. 2002.